SEQUENT

DYNIX/ptx Reference Manual
Section 1, M-Z

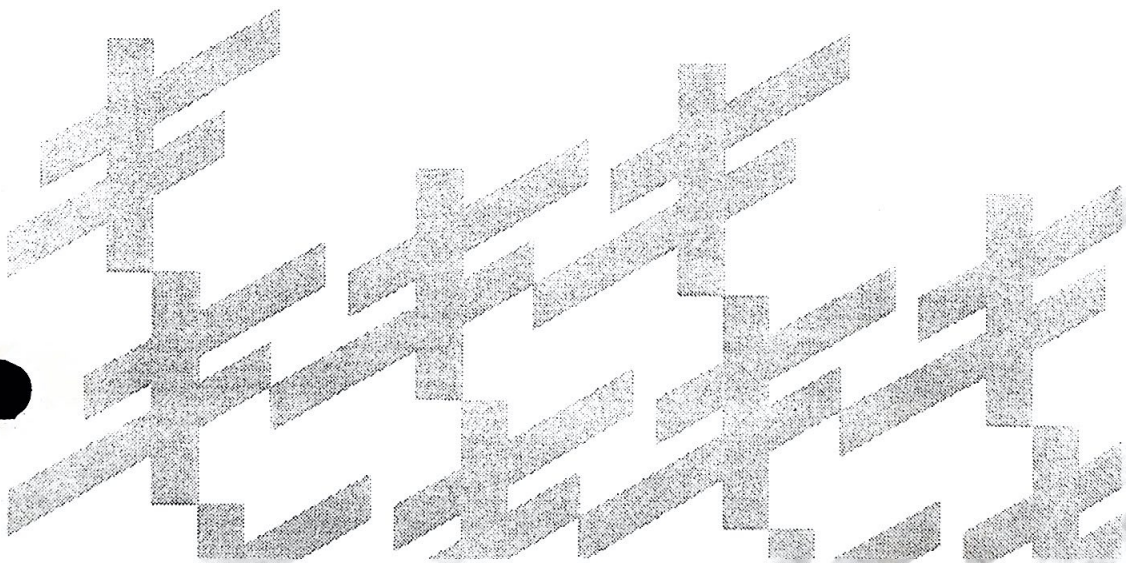
1003-49231-01

THIS SHEET LIFTER
MUST BE PLACED ON
TOP OF CONTENTS

GRAPHIC PRODUCTS
Chicago, Illinois 60609

DYNIX/ptx®
Reference Manual
Section 1, M-Z

1003-49231-01
Sequent Computer Systems, Inc.



ACT is a trademark of Micro-Term.
Ann Arbor is a trademark of Ann Arbor Terminals.
Beehive is a trademark of Beehive International.
Concept is a trademark of Human Designed Systems.
Dataphone is a registered trademark of AT&T.
DEC is a trademark of Digital Equipment.
Diablo is a registered trademark of Xerox.
Digital is a trademark of Digital Equipment.
DOCUMENTER'S WORKBENCH is a trademark of AT&T.
DYNIX and DYNIX/ptx are registered trademarks of Sequent Computer Systems, Inc.
HP is a trademark of Hewlett-Packard.
i386 is a trademark of Intel Corp.
LSI is a trademark of Lear Siegler.
Micro-term and MIME are trademarks of Micro-Term.
MULTIBUS is a trademark of Intel Corp.
PDP is a trademark of Digital Equipment.
Sequent is a trademark of Sequent Computer Systems, Inc.
Symmetry is a registered trademark of Sequent Computer Systems, Inc.
Tektronix and Tektronix 4010 are registered trademarks of Tektronix.
Teleray is a trademark of Research.
Teletype is a registered trademark of AT&T.
TeleVideo is a registered trademark of TeleVideo Systems.
UNIX is a registered trademark of AT&T.
VAX is a trademark of Digital Equipment.
Versatec is a registered trademark of Versatec.
VT100 is a trademark of Digital Equipment.

Copyright © 1989, 1990 by Sequent Computer Systems, Inc. All rights reserved. This document may not be copied or reproduced in any form without permission from Sequent Computer Systems, Inc. Information in this document is subject to change without notice.

Printed in the United States of America.

About This Manual

Overview

The *DYNIX/ptx Reference Manual* is for programmers working in the DYNIX/ptx operating system environment. This manual describes the programming features of the operating system and assumes that the audience is familiar with the operating system. It does not provide a general overview of the operating system or implementation details of the system.

Organization

The *DYNIX/ptx Reference Manual* is contained in five binders organized as follows:

- Section 1 A-F **Commands**
- Section 1 G-Z **Commands**
- Section 2 **System Calls**
  Section 2SEQ **Sequent System Calls**
  Section 3 **Subroutines**
    - Section 3C **C Library Routines**
    - Section 3M **Mathematical Library Routines**
    - Section 3N **Networking Support Utilities**
    - Section 3PPS **Sequent Parallel Programming Library**
    - Section 3S **Standard I/O Library Routines**
    - Section 3SEQ **Sequent I/O Library Routines**
    - Section 3X **Special Routines**
- Section 4 **File Formats**
  Section 5 **Miscellaneous Facilities**
- Section 7 **Special Files**
  Section 8 **Stand-alone Utilities**

Each binder also contains the following items:

- **Table of Contents** that alphabetically lists the contents of the entire manual set
- **Permuted Index** that lists each programming feature in the entire manual set

The permuted index is a list of keywords, given alphabetically in the second of three columns, together with the context in which each keyword is found. The following shows four separate entries illustrating four ways that the **cat** command can be looked up in the permuted index. Note that you can find the **cat** command by looking up "cat," "concatenate," "files," or "print."

```
                    cat: concatenate and print files.....cat(1)
               cat: concatenate and print files..... cat(1)
concatenate and print files.....cat(1)
     concatenate and print files..... cat(1)
```

Keywords are either topical keywords or the names of manual entries themselves. Entries are identified with their section numbers shown in parentheses. The numbers are important because there is duplication of names among the sections. The right column lists the name of the manual page on which each keyword may be found. The left column contains useful information about the keyword.

The following paragraphs briefly describe the other sections of the manual.

Section 1: Commands describes operating system commands and commands that support C and other programming languages.

Section 2: System Calls describes the access to the services provided by the operating system kernel, including the C language interface.

Section 2SEQ: Sequent System Calls describes the access to the Sequent services provided by the operating system kernel, including the C language interface.

Section 3: Subroutines describes the available subroutines. Their binary versions reside in various system libraries in the directories */lib* and */usr/lib*. The introduction to Section 3 describes these libraries and the files in which they are stored.

Section 4: File Formats documents the structure of particular kinds of files; for example, the format of the output of the link editor is given in *a.out*(4). Excluded are files used by only one command (for example, the assembler's intermediate files). In general, the C language structures corresponding to these formats can be found in the directories */usr/include* and */usr/include/sys*.

Section 5: Miscellaneous Facilities contains a variety of descriptions including descriptions of character sets, macro packages, and so forth.

Section 7: Special Files describes various special files that refer to specific hardware peripherals and operating system device drivers.

Section 8: Stand-alone Utilities describes stand-alone procedures.

References to manual pages with numbers other than those above mean that the utility is contained in the appropriate section of another manual. Each section consists of independent entries which are alphabetized, with the exception of the introductions that begin some sections. Section 3 is divided into subsections in alphabetical order by suffix. Some entries may describe several routines, commands, and so forth. In such cases, the entry appears only once, alphabetized under its *primary* name, the name that appears at the upper corners of each manual page.

Each manual page can be viewed online with the **man**(1) command. For example, enter **man shell** to view the **shell** manual page. Enter **man man** for more information about the **man** command.

Notational Conventions

Manual pages are based on a common format. The main headings are described here—not all of them appear in all cases:

- **NAME** gives the name(s) of the entry and briefly states its purpose. This entry is used to create the Permuted Index and the Table of Contents.
- **SYNOPSIS** summarizes the usage of the program being described. Conventions vary in some sections but are generally as follows:

- **Boldface** strings are literals and are to be typed just as they appear.
- *Italic* strings usually represent substitutable argument prototypes and program names found elsewhere in the manual.
- Square brackets [ ] around an argument prototype indicate that the argument is optional. When an argument prototype is given as *name* or *file*, it refers to a *filename*.
- Ellipses ... are used to show that the previous argument prototype may be repeated.
- A final convention is used by the commands themselves. An argument beginning with a minus –, plus +, or equal sign = is often taken to be some sort of flag argument even if it appears in a position where a file name could appear. Therefore, it is unwise to have files whose names begin with –, +, or =.

- **CONFIGURATION ENTRY** shows how a driver is called out in a system configuration file.
- **DESCRIPTION** describes the entry, including its purpose and how it is used.
- **FIELDS** describe system file or record entries.
- **EXAMPLES** and **BINARY CONFIGURATION** give examples of usage.
- **FILES** list files that are part of a program, created by a program, or related to a program.
- **SEE ALSO** points to related information in other manual pages.
- **DIAGNOSTICS** discuss the diagnostic (error) messages that may be produced. Messages that are self-explanatory are not listed.
- **WARNINGS** contains warnings against improper user actions.
- **NOTES** give hints about the use of the utility.
- **BUGS** describe known bugs in the software.

Contents

1 Commands

intro(1) ..... introduction to maintenance commands and application programs
300(1): 300, 300s ..... handle special functions of DASI 300 and 300s terminals
4014(1) ..... paginator for the Tektronix 4014 terminal
450(1) ..... handle special functions of the DASI 450 terminal
accept(1M): accept, reject ..... allow or prevent LP requests
acct(1M): acctdisk, acctdusg, accton, acctwtmp ..... overview of accounting and miscellaneous accounting commands
acctcms(1M) ..... command summary from per-process accounting records
acctcom(1) ..... search and print process accounting files
acctcon(1M): acctcon1, acctcon2 ..... connect-time accounting
acctmerg(1M) ..... merge or add total accounting files
acctprc(1M): acctprc1, acctprc2 ..... process accounting
acctsh(1M): chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, shutacct, startup, turnacct ..... shell procedures for accounting
addcmdopt(1M) ..... add a command option to a menu
addformopt(1M) ..... add a form/SELECT window option to a menu
addmenuopt(1M) ..... add a menu option to a menu
admin(1) ..... create and administer SCCS files
apropos(1) ..... locate commands by keyword lookup
ar(1) ..... see *C and Language Tools Binder*
as(1) ..... see *C and Language Tools Binder*
asa(1) ..... interpret ASA carriage control characters
assist(1) ..... assistance using operating system commands
astgen(1) ..... program for generating/modifying ASSIST menus or command forms
at(1): at, batch ..... execute commands at a later time
awk(1) ..... pattern scanning and processing language
banner(1) ..... make posters
basename(1): basename, dirname ..... deliver portions of pathnames
bc(1) ..... arbitrary-precision arithmetic language
bcheckrc(1M) ..... system initialization procedures
bdflush(1M) ..... update daemon
bdiff(1) ..... big diff
bfs(1) ..... big file scanner
bomverify(1M) ..... perform an installation consistency check based on the BOM
bootflags(1M) ..... display/change system boot flags
bp(1M) ..... binary patcher
cal(1) ..... print calendar

calendar(1) ..... reminder service
captoinfo(1M) ..... convert a termcap description into a terminfo description
cat(1) ..... concatenate and print files
cb(1) ..... see *C and Language Tools Binder*
cc(1) ..... see *C and Language Tools Binder*
cd(1) ..... change working directory
cdc(1) ..... change the delta commentary of an SCCS delta
cflow(1) ..... see *C and Language Tools Binder*
checkeq(1) ..... check documents for correct eqn syntax.
checkfsys(1M) ..... check a filesystem
chmod(1) ..... change mode
chown(1): chown, chgrp ..... change owner or group
chroot(1M) ..... change root directory for a command
chrtbl(1M) ..... generate character classification and conversion tables
ci(1) ..... check in RCS revisions
ckbupscd(1M) ..... check filesystem backup schedule
clri(1M) ..... clear inode
cmp(1) ..... compare two files
co(1) ..... check out RCS revisions
col(1) ..... filter reverse line-feeds
comb(1) ..... combine SCCS deltas
comm(1) ..... select or reject lines common to two sorted files
compress(1): compress,
uncompress, zcat ..... compress and expand data
config(1M) ..... build system configuration files
coproc(1F): cocreate, cosend,
cocheck, coreceive, codestroy ..... communicate to a process
cp(1): cp, ln, mv ..... copy, link or move files
cpp(1) ..... see *C and Language Tools Binder*
cpio(1) ..... copy file archives in and out
cprs(1) ..... see *C and Language Tools Binder*
cpset(1) ..... copy file and set mode, user, and group IDs
crash(1M) ..... examine system images
cron(1M) ..... clock daemon
crontab(1) ..... user crontab file
crypt(1) ..... encode/decode
csh(1) ..... a shell (command interpreter) with C-like syntax
csplit(1) ..... context split
ct(1C) ..... spawn getty to a remote terminal
ctags(1) ..... create a tags file
ctrace(1) ..... see *C and Language Tools Binder*
cu(1C) ..... call another UNIX system
cut(1) ..... cut out selected fields of each line of a file
cxref(1) ..... see *C and Language Tools Binder*
date(1) ..... print and set the date
dbx(1) ..... see *C and Language Tools Binder*
dc(1) ..... desk calculator
dcheck(1M) ..... filesystem directory consistency check
dd(1M) ..... convert and copy a file
delta(1) ..... make a delta (change) to an SCCS file

deroff(1) ..... remove nroff, tbl, and eqn constructs
devbuild(1M): devbuild,
devdestroy ..... build and destroy virtual devices
devnm(1M) ..... device name
df(1M) ..... report number of free disk blocks and inodes
diff(1) ..... differential file comparison
diff3(1) ..... 3-way differential file comparison
dircmp(1) ..... directory comparison
dis(1) ..... see *C and Language Tools Binder*
diskusg(1M) ..... generate disk accounting data by user ID
dltsectbl(1M) ..... refresh security table
dmesg(1M) ..... collect system diagnostic messages to form error log
du(1M) ..... summarize disk usage
dump(1) ..... see *C and Language Tools Binder*
dump(1M) ..... incremental filesystem dump
dumpconf(1M) ..... dump kernel auto-configuration device table
dumpfs(1M) ..... dump filesystem information
ec(1M) ..... forms compiler
echo(1) ..... echo arguments
echo(1F) ..... put string on virtual output
ed(1): ed, red ..... text editor
edc(1M) ..... run an externally described command
edit(1) ..... text editor (variant of ex for casual users)
edquota(1M) ..... edit user quotas
efl(1) ..... Extended Fortran Language
egrep(1) ..... search a file for a pattern using full regular expressions
enable(1): enable, disable ..... enable/disable LP printers
env(1) ..... set environment for command execution
ex(1) ..... text editor
expr(1) ..... evaluate arguments as an expression
factor(1) ..... obtain the prime factors of a number
ff(1M) ..... list filenames and statistics for a filesystem
fgrep(1) ..... search a file for a character string
file(1) ..... determine file type
find(1) ..... find files
format(1M) ..... format disks or perform other defect management functions
fsck(1M) ..... filesystem consistency check and interactive repair
fsirand(1M) ..... install random inode generation numbers
fsplit(1) ..... split f77, ratfor, or efl files
fsstat(1M) ..... report filesystem status
fstyp(1M) ..... determine filesystem identifier
fuser(1M) ..... identify processes using a file or file structure
fwtmp(1M): fwtmp, wtmpfix ..... manipulate connect accounting records
gdev(1G): hpd, erase, hardcopy,
tekset, td ..... graphical device routines and filters
ged(1G) ..... graphical editor
gencc(1M) ..... create a front-end to the cc command
get(1) ..... get a version of an SCCS file
getfrm(1F) ..... returns the current frame number

getitems(1F) ..... returns a list of the currently marked menu items
getopt(1) ..... parse command options
getopts(1): getopts, getoptcvt ..... parse command options
getty(1M) ..... set terminal type, modes, speed, and line discipline
glossary(1) ..... definitions of common operating system terms and symbols
graph(1G) ..... draw a graph
graphics(1G) ..... access graphical and numerical commands
greek(1) ..... select terminal filter
grep(1) ..... search a file for a pattern
groups(1) ..... show group memberships
gutil(1G) ..... graphical utilities
help(1) ..... operating system Help Facility
helpadm(1M) ..... change the Help Facility database
hp(1) ..... handle special functions of Hewlett-Packard terminals
hpio(1) ..... Hewlett-Packard 2645A terminal tape file archiver
icheck(1M) ..... filesystem storage consistency check
id(1M) ..... print user and group IDs and names
ident(1) ..... identify files
indicator(1F) ..... display application specific alarms and/or the working indicator
infocmp(1M) ..... compare or print out terminfo descriptions
init(1M): init, telinit ..... process control initialization
install(1M) ..... install commands
installboot(1M) ..... installs a boot program on a disk
ipcrm(1) ..... remove a message queue, semaphore set, or shared memory ID
ipcs(1) ..... report inter-process communication facilities status
isismgr(1M) ..... ISIS file manager, updates text files in ISIS file format
join(1) ..... relational database operator
kill(1) ..... terminate a process
killall(1M) ..... kill all active processes
ksh(1): ksh, rksh ..... the KornShell, a standard/restricted command and programming language
labelit(1M) ..... provide labels for filesystems
ld(1) ..... see *C and Language Tools Binder*
lex(1) ..... generate programs for simple lexical tasks
line(1) ..... read one line
link(1M): link, unlink ..... link and unlink files and directories
lint(1) ..... see *C and Language Tools Binder*
list(1) ..... see *C and Language Tools Binder*
locate(1) ..... use keywords to identify an operating system command
login(1) ..... sign on
logname(1) ..... get login name
lorder(1) ..... see *C and Language Tools Binder*
lp(1): lp, cancel ..... send/cancel requests to the LP print service
lpadmin(1M) ..... configure the LP print service
lpfilter(1M) ..... administer filters used with the LP print service
lpforms(1M) ..... administer forms used with the LP print service

lpsched(1M): lpsched, lpshut,
lpmove ..... start/stop the LP print service and move requests
lpstat(1) ..... print information about the status of the LP print service
lpusers(1M) ..... set printing queue priorities
ls(1) ..... list contents of directory
m4(1) ..... macro processor
machid(1): i386, ns32000, pdp11,
u3b, u3b2, u3b5, vax ..... get processor type truth value
mail(1): mail, rmail ..... send mail to users or read mail
mailbug(1) ..... submit a bug report or enhancement request
mailx(1) ..... interactive message processing system
make(1) ..... maintain, update, and regenerate groups of programs
makealiases(1M): makealiases,
makewhatis ..... generate man page databases
makedev(1M) ..... make system special files
makekey(1) ..... generate encryption key
man(1) ..... display on-line man pages
mcs(1) ..... manipulate the object file comment section
menu(1M) ..... menu processor
menumsg(1M) ..... route user message to menu message queue
merge(1) ..... three-way file merge
mesg(1) ..... permit or deny messages
message(1F) ..... puts its arguments on message line
mkdir(1) ..... make directories
mkfs(1M) ..... construct a filesystem
mknod(1M) ..... build special file
mkpwdbm(1M) ..... make a database for fast password lookup
mkvtoc(1M) ..... populate VTOC on disks
monitor(1M) ..... monitor system activity
more(1) ..... view a file one full screen at a time
mount(1M): mount, umount ..... mount and unmount filesystems
mountall(1M): mountall,
umountall ..... mount, unmount multiple filesystems
mt(1) ..... magnetic tape manipulating program
mvdir(1M) ..... move a directory
nawk(1) ..... pattern scanning and processing language
ncheck(1M) ..... generate pathnames from i-numbers
newform(1) ..... change the format of a text file
newfs(1M) ..... construct a new file system
newgrp(1M) ..... log in to a new group
news(1) ..... print news items
nice(1) ..... run a command at low priority
nl(1) ..... line numbering filter
nm(1) ..... see *C and Language Tools Binder*
noage(1M): noage, nopff, noswap,
onproc ..... run a command with priority aging, PFF, swapping, or processor migration disabled
nohup(1) ..... run a command immune to hangups and quits
od(1) ..... octal dump
online(1M): online, offline ..... bring processors online and offline

pack(1): pack, pcat, unpack ..... compress and expand files
passmgmt(1M) ..... password files management
passwd(1) ..... change login password and password attributes
paste(1) ..... merge same lines of several files or subsequent lines of one file
pathconv(1F) ..... search Interpreter criteria for filename
pdbx(1) ..... see *C and Language Tools Binder*
pg(1) ..... file perusal filter for CRTs
pmap(1M) ..... display or alter parameters of virtual-to-physical mapping devices
pr(1) ..... print files
prof(1) ..... display profile data
prompt(1M) ..... prompt program
prs(1) ..... print an SCCS file
prtvtoc(1M) ..... print the VTOC of a block device
ps(1) ..... report process status
pwck(1M): pwck, grpck ..... password/group file checkers
pwconv(1M) ..... installs and updates /etc/shadow with information from /etc/passwd
pwd(1) ..... working directory name
pwdhd(1M) ..... daemon to oversee password database creation
pwfix(1M) ..... remove incorrectly formatted password entries
pwunconv(1M) ..... converts from a two to a one password file scheme
quota(1) ..... display disk usage and limits
quotacheck(1M) ..... filesystem quota consistency checker
quotaon(1M): quotaon, quotaoff ..... turn filesystem quotas on and off
ratfor(1) ..... rational Fortran dialect
rc0(1M) ..... run commands performed to stop the operating system
rc2(1M) ..... run commands performed for multi-user environment
rcs(1) ..... change RCS file attributes
rcsdiff(1) ..... compare RCS revisions
rcsintro(1) ..... introduction to RCS commands
rcsmerge(1) ..... merge RCS revisions
readfile(1F): readfile, longline ..... reads file and gets longest line
regcmp(1) ..... regular expression compile
regex(1F) ..... match patterns against a string
reinit(1F) ..... run an initialization file
repquota(1M) ..... summarize quotas for a filesystem
reset(1F) ..... reset the current form field to its default values
restore(1M) ..... incremental filesystem restore
rlog(1) ..... print log messages and other information about RCS files
rm(1): rm, rmdir ..... remove files or directories
rmdel(1) ..... remove a delta from an SCCS file
rmvopt(1M) ..... remove a menu option
run(1F) ..... run an executable
runacct(1M) ..... run daily accounting
sact(1) ..... print current SCCS file editing activity
sag(1G) ..... system activity graph
sar(1) ..... system activity reporter
sar(1M): sa1, sa2, sadc ..... system activity report package

savecore(1M) ..... save a core dump of the operating system
sccsdiff(1) ..... compare two versions of an SCCS file
sccstorcs(1) ..... build RCS file from SCCS file
sdiff(1) ..... side-by-side difference program
sed(1) ..... stream editor
set(1F): set, unset ..... set and unset environment variables in core or in files
setcolor(1F) ..... redefine or create a color
setmnt(1M) ..... establish mount table
setpgrp(1) ..... run command in its own process group
setup(1) ..... initialize system for first user
sh(1): sh, rsh ..... shell, the standard/restricted command programming language
shell(1F) ..... run a command using shell
showcfg(1M) ..... show the configuration of the machine
shutdown(1M) ..... shut down system, change system state
size(1) ..... see *C and Language Tools Binder*
sleep(1) ..... suspend execution for an interval
sort(1) ..... sort and/or merge files
spell(1): spell, hashmake, spellin, hashcheck ..... find spelling errors
spline(1G) ..... interpolate smooth curve
split(1) ..... split a file into pieces
starter(1) ..... information about the operating system for beginning users
stat(1G) ..... statistical network useful with graphical commands
strace(1M) ..... print STREAMS trace messages
strclean(1M) ..... STREAMS error logger cleanup program
strerr(1M) ..... STREAMS error logger daemon
strip(1) ..... see *C and Language Tools Binder*
stty(1) ..... set the options for a terminal
su(1M) ..... become superuser or another user
sum(1) ..... print checksum and block count of a file
swap(1M) ..... swap administrative interface
sweepon(1M): sweepon, sweepoff ..... enable/disable patrol seeks on DCC disks
sync(1M) ..... update the superblock
sysadm(1) ..... menu interface to do system administration
tabs(1) ..... set tabs on a terminal
tail(1) ..... deliver the last part of a file
tar(1) ..... tape file archiver
tasktbl(1M) ..... create security table in shared memory
tee(1) ..... pipe fitting
test(1) ..... condition evaluation command
tfipc(1M) ..... interprocess communication reporter
tfmenush(1M) ..... assign menu system as user login shell
tfnotify(1M) ..... send completion message
tic(1M) ..... terminfo compiler
time(1): time, ptime ..... time a command
timex(1) ..... time a command; report process data and system activity
toc(1G): dtoc, ttoc, vtoc ..... graphical table of contents routines
touch(1) ..... update access and modification times of a file

tplot(1G) ..... graphics filters
tput(1) ..... initialize a terminal or query terminfo database
tr(1) ..... translate characters
true(1): true, false ..... provide truth values
tsort(1) ..... topological sort
tty(1) ..... get the name of the terminal
ttydevname(1) ..... get the device associated with a terminal or the terminal associated with a device
tunefs(1M) ..... tune up an existing filesystem
uadmin(1M) ..... administrative control
umask(1) ..... set file-creation mode mask
uname(1) ..... print name of current system
unget(1) ..... undo a previous get of an SCCS file
uniq(1) ..... report repeated lines in a file
units(1) ..... conversion program
uptime(1) ..... show how long system has been up
usage(1) ..... retrieve a command description and usage examples
uucheck(1M) ..... check the uucp directories and permissions file
uucico(1M) ..... file transport program for the uucp system
uucleanup(1M) ..... uucp spool directory clean-up
uucp(1C): uucp, uulog, uuname ..... UNIX-to-UNIX system copy
uugetty(1M) ..... set terminal type, modes, speed, and line discipline
uusched(1M) ..... the scheduler for the uucp file transport program
uustat(1C) ..... uucp status inquiry and job control
uuto(1C): uuto, uupick ..... public UNIX-to-UNIX system file copy
uutry(1M) ..... try to contact remote system with debugging on
uux(1C) ..... UNIX-to-UNIX system command execution
uuxqt(1M) ..... execute remote command requests
val(1) ..... validate SCCS file
vc(1) ..... version control
vi(1) ..... screen-oriented (visual) display editor based on ex
vmtune(1M) ..... tune system-wide virtual-memory parameters
volcopy(1M) ..... make literal copy of filesystem
vsig(1F) ..... synchronize a co-process with the controlling FMLI object
wait(1) ..... await completion of process
wall(1) ..... write to all users
wc(1) ..... word count
what(1) ..... identify SCCS files
whatis(1) ..... describe what a command is
who(1) ..... who is on the system
whodo(1M) ..... who is doing what
write(1) ..... write to another user
xargs(1) ..... construct argument list(s) and execute command
yacc(1) ..... yet another compiler-compiler

2 System Calls

intro(2) ..... introduction to system calls and error numbers
access(2) ..... determine accessibility of a file
acct(2) ..... enable or disable process accounting
alarm(2) ..... set a process alarm clock
brk(2): brk, sbrk ..... change data segment space allocation
chdir(2) ..... change working directory
chmod(2) ..... change mode of file
chown(2) ..... change owner and group of a file
chroot(2) ..... change root directory
close(2) ..... close a file descriptor
creat(2) ..... create a new file or rewrite an existing one
dup(2) ..... duplicate an open file descriptor
dup2(2) ..... duplicate an open file descriptor
exec(2): execl, execv, execle,
execve, execlp, execvp ..... execute a file
exit(2): exit, _exit ..... terminate process
fcntl(2) ..... file control
fork(2) ..... create a new process
getdents(2) ..... read directory entries and put in a file system independent format
getgroups(2) ..... get supplementary group IDs
getmsg(2) ..... get next message off a stream
getpid(2): getpid, getpgrp, getppid .... get process, process group, and parent process IDs
getuid(2): getuid, geteuid, getgid,
getegid ..... get real user, effective user, real group, and effective group IDs
ioctl(2) ..... control device
kill(2) ..... send a signal to a process or a group of processes
link(2) ..... link to a file
lseek(2) ..... move read/write file pointer
mkdir(2) ..... make a directory
mknod(2): mknod ..... make a directory, or a special or ordinary file
mount(2) ..... mount a filesystem
msgctl(2) ..... message control operations
msgget(2) ..... get message queue
msgop(2) ..... message operations
nice(2) ..... change priority of a process
open(2) ..... open for reading or writing
pathconf(2): pathconf, fpathconf ..... get configurable pathname variables
pause(2) ..... suspend process until signal
pipe(2) ..... create an interprocess channel
plock(2) ..... lock process, text, or data in memory
poll(2) ..... STREAMS input/output multiplexing
profil(2) ..... execution time profile
ptrace(2): ptrace, mptrace ..... process trace facility
putmsg(2) ..... send a message on a stream
read(2) ..... read from file
rename(2) ..... change the name of a file
rmdir(2) ..... remove a directory

semctl(2) ..... semaphore control operations
semget(2) ..... get set of semaphores
semop(2) ..... semaphore operations
setpgid(2) ..... manipulate signal sets
setpgrp(2) ..... set process group ID
setsid(2) ..... create session and set process group ID
setuid(2): setuid, setgid ..... set user and group IDs
shmctl(2) ..... shared memory control operations
shmget(2) ..... get shared memory segment identifier
shmop(2): shmop, shmat, shmdt ..... shared memory operations
sigaction(2) ..... examine and change signal actions
sigemptyset(2): sigemptyset,
sigfillset, sigaddset, sigdelset,
sigismember ..... manipulate signal sets
signal(2) ..... specify what to do upon receipt of a signal
sigpending(2) ..... examine pending signals
sigprocmask(2) ..... examine and change blocked signals
sigset(2): sigset, sighold, sigrelse,
sigignore, sigpause ..... signal management
sigsuspend(2) ..... wait for a signal
stat(2): stat, lstat, fstat ..... get file status
statfs(2): statfs, fstatfs ..... get filesystem information
stime(2) ..... set time
stty(2): stty, gtty ..... set and get terminal state (defunct)
sync(2) ..... update superblock
syscall(2) ..... indirect system call
sysconf(2) ..... get configurable system variables
sysi86(2) ..... machine specific functions
tcgetattr(2): tcgetattr, tcsetattr ..... get/set terminal state
tcgetpgrp(2) ..... get foreground process group ID
tcsetpgrp(2) ..... set foreground process group ID
time(2) ..... get time
times(2) ..... get process and child process times
uadmin(2) ..... administrative control
ulimit(2) ..... get and set user limits
umask(2) ..... set and get file creation mask
umount(2) ..... unmount a filesystem
uname(2) ..... get name of current operating system
unlink(2) ..... remove directory entry
ustat(2) ..... get filesystem statistics
utime(2) ..... set file access and modification times
vfork(2) ..... spawn new process in a virtual memory efficient way
wait(2): wait, waitpid ..... wait for a child process to stop or terminate
write(2) ..... write on a file

2SEQ Sequent System Calls

intro(2SEQ) ..... introduction to Sequent system calls
bdflush(2SEQ) ..... update daemon
fvhangup(2SEQ) ..... virtually hangup the specified controlling terminal
get_process_stats(2SEQ) ..... get resource utilization information
getdtablemax(2SEQ) ..... get maximum in-use descriptor table index
getdtablesize(2SEQ) ..... get descriptor table size
getpagesize(2SEQ) ..... get system page size
mmap(2SEQ) ..... map an open file into the processs address space
munmap(2SEQ) ..... unmap part of the processs address space
proc_ctl(2SEQ) ..... control over various process attributes
quotactl(2SEQ) ..... manipulate disk quotas
readlink(2SEQ) ..... read value of a symbolic link
readv(2SEQ) ..... read vector from file
setdtablesize(2SEQ) ..... set descriptor table size
setgroups(2SEQ) ..... set group access list
sigcontext(2SEQ) ..... get signal context
sigstack(2SEQ) ..... set and/or get signal stack context
symlink(2SEQ) ..... make symbolic link to a file
tmp_affinity(2SEQ) ..... bind a process to a processor
tmp_ctl(2SEQ) ..... multi-processor control
truncate(2SEQ): truncate,
ftruncate ..... truncate a file to a specified length
vm_ctl(2SEQ) ..... examine and change virtual memory tuning parameters
writev(2SEQ) ..... write vector on a file

3 Subroutines

intro(3) ..... introduction to functions and libraries

3C C Library Routines

a64l(3C): a64l, l64a ..... convert between long integer and base-64 ASCII string
abort(3C) ..... generate an IOT fault
abs(3C) ..... return integer absolute value
bsearch(3C) ..... binary search a sorted table
clock(3C) ..... report CPU time used
conv(3C): toupper, tolower,
_toupper, _tolower, toascii ..... translate characters
crypt(3C): crypt, setkey, encrypt ..... generate hashing encryption
ctime(3C): ctime, localtime,
gmtime, asctime, cftime, ascftime,
mktime, tzset ..... convert date and time to string
ctype(3C): isdigit, isxdigit,
islower, isupper, isalpha, isalnum,
isspace, iscntrl, ispunct, isprint,
isgraph, isascii, tolower, toupper,
toascci, _tolower, _toupper,
setchrclass ..... character handling

dev_to_ttyname(3C): dev_to_ttyname, ttyname_to_dev ..... convert between major/minor numbers and tty names
dial(3C) ..... establish an out-going terminal line connection
drand48(3C): drand48, erand48, lrand48, nrand48, mrand48, jrand48, srand48, seed48, lcong48 ..... generate uniformly distributed pseudorandom numbers
ecvt(3C): ecvt, fcvt, gcvt ..... convert floating-point number to string
end(3C): end, etext, edata ..... last locations in program
fpgetround(3C): fpgetround, fpsetround, fpgetmask, fpsetmask, fpgetsticky, fpsetsticky ..... IEEE floating-point environment control
frexp(3C): frexp, ldexp, modf ..... manipulate parts of floating-point numbers
ftw(3C) ..... walk a file tree
getcwd(3C) ..... get pathname of current working directory
getenv(3C) ..... return value for environment name
getgrent(3C): getgrent, getgrgid, getgrnam, setgrent, endgrent, fgetgrent ..... get group file entry
getlogin(3C) ..... get login name
getopt(3C) ..... get option letter from argument vector
getpass(3C) ..... read a password
getpw(3C) ..... get name from UID
getpwent(3C): getpwent, getpwuid, getpwnam, setpwent, endpwent, fgetpwent ..... get password file entry
getut(3C): getutent, getutid, getutline, pututline, setutent, endutent, utmpname, getutpid ..... access utmp file entry
hsearch(3C): hsearch, hcreate, hdestroy ..... manage hash search tables
isnan(3C): isnand, isnanf ..... test for floating-point NaN (Not-A-Number)
l3tol(3C): l3tol, ltol3 ..... convert between 3-byte integers and long integers
lockf(3C) ..... record locking on files
lsearch(3C): lsearch, lfind ..... linear search and update
malloc(3C): malloc, free, realloc, calloc ..... main memory allocator
memory(3C): memccpy, memchr, memcmp, memcpy, memset ..... memory operations
mkfifo(3C) ..... make a FIFO special file
mktemp(3C) ..... make a unique filename
monitor(3C) ..... prepare execution profile
nlist(3C) ..... get entries from name list
perror(3C): perror, errno, sys_errlist, sys_nerr ..... system error messages
putenv(3C) ..... change or add value to environment
putpwent(3C) ..... write password file entry
qsort(3C) ..... quicker sort
rand(3C): rand, srand ..... simple random-number generator
setjmp(3C): setjmp, longjmp ..... non-local goto
setlocale(3C) ..... set programs location

sigsetjmp(3C): sigsetjmp,
siglongjmp ..... non-local jumps
sleep(3C) ..... suspend execution for interval
ssignal(3C): ssignal, gsignal ..... software signals
stdipc(3C) ..... standard interprocess communication package
strftime(3C) ..... format time information
string(3C): strcat, strdup, strncat,
strcmp, strncmp, strcpy, strncpy,
strlen, strchr, strrchr, strpbrk,
strspn, strcspn, strtok ..... string operations
strstr(3C) ..... find first occurrence of character
strtod(3C): strtod, atof ..... convert string to double-precision number
strtol(3C): strtol, atol, atoi ..... convert string to integer
swab(3C) ..... swap bytes
tsearch(3C): tsearch, tfind,
tdelete, twalk ..... manage binary search trees
ttyname(3C): ttyname, isatty ..... find name of a terminal
ttyslot(3C) ..... find the slot in the utmp file of the current user

3M Mathematical Library Routines

bessel(3M): j0, j1, jn, y0, y1, yn ..... Bessel functions
erf(3M): erf, erfc ..... error function and complementary error function
exp(3M): exp, log, log10, pow, sqrt .... exponential, logarithm, power, square root functions
floor(3M): floor, ceil, fmod, fabs ..... floor, ceiling, remainder, absolute value functions
gamma(3M) ..... log gamma function
hypot(3M) ..... Euclidean distance function
matherr(3M) ..... error-handling function
sinh(3M): sinh, cosh, tanh ..... hyperbolic functions
trig(3M): sin, cos, tan, asin, acos,
atan, atan2 ..... trigonometric functions

3N Networking Support Utilities

t_accept(3N) ..... accept a connect request
t_alloc(3N) ..... allocate a library structure
t_bind(3N) ..... bind an address to a transport endpoint
t_close(3N) ..... close a transport endpoint
t_connect(3N) ..... establish a connection with another transport user
t_error(3N) ..... produce error message
t_free(3N) ..... free a library structure
t_getinfo(3N) ..... get protocol-specific service information
t_getstate(3N) ..... get the current state
t_listen(3N) ..... listen for a connect request
t_look(3N) ..... look at the current event on a transport endpoint
t_open(3N) ..... establish a transport endpoint
t_optmgmt(3N) ..... manage options for a transport endpoint
t_rcv(3N) ..... receive data or expedited data sent over a connection
t_rcvconnect(3N) ..... receive the confirmation from a connect request
t_rcvdis(3N) ..... retrieve information from disconnect
t_rcvrel(3N) ..... acknowledge receipt of an orderly release indication

t_rcvudata(3N) ..... receive a data unit
t_rcvuderr(3N) ..... receive a unit data error indication
t_snd(3N) ..... send data or expedited data over a connection
t_snddis(3N) ..... send user-initiated disconnect request
t_sndrel(3N) ..... initiate an orderly release
t_sndudata(3N) ..... send a data unit
t_sync(3N) ..... synchronize transport library
t_unbind(3N) ..... disable a transport endpoint

3PPS Sequent Parallel Programming Library

intro(3PPS) ..... introduction to Parallel Programming Library
brk(3PPS): brk, sbrk ..... change private data segment size
cpus_online(3PPS) ..... returns the number of CPUs on-line
m_fork(3PPS) ..... execute a subprogram in parallel
m_get_myid(3PPS) ..... return process identification
m_get_numprocs(3PPS) ..... return number of child processes
m_kill_procs(3PPS) ..... kill child processes
m_lock(3PPS): m_lock, m_unlock ..... lock, unlock locks
m_next(3PPS) ..... increment global counter
m_park_procs(3PPS): m_park_procs,
m_rele_procs ..... suspend and resume child process execution
m_set_procs(3PPS) ..... set number of child processes
m_single(3PPS): m_single,
m_multi ..... mark single-process code section
m_sync(3PPS) ..... check in at barrier
s_lock(3PPS): s_init_lock, s_lock,
s_clock, s_unlock ..... initialize, lock, unlock locks
s_wait_barrier(3PPS): s_init_barrier,
s_wait_barrier ..... initialize barrier, wait at barrier
shbrk(3PPS): shbrk, shsbrk ..... change shared data segment size
shmalloc(3PPS): shmalloc,
shrealloc, shfree ..... shared memory allocator

3S Standard I/O Library Routines

ctermid(3S) ..... generate filename for terminal
cuserid(3S) ..... get character login name of the user
fclose(3S): fclose, fflush ..... close or flush a stream
ferror(3S): ferror, feof, clearerr,
fileno ..... stream status inquiries
fopen(3S): fopen, freopen, fdopen ..... open a stream
fread(3S): fread, fwrite ..... binary input/output
fseek(3S): fseek, rewind, ftell ..... reposition a file pointer in a stream
getc(3S): getc, getchar, fgetc, getw .... get character or word from a stream
gets(3S): gets, fgets ..... get a string from a stream
popen(3S): popen, pclose ..... initiate pipe to/from a process
printf(3S): printf, fprintf, sprintf ..... print formatted output

putc(3S): putc, putchar, fputc,
putw ..... put character or word on a stream
puts(3S): puts, fputs ..... put a string on a stream
scanf(3S): scanf, fscanf, sscanf ..... convert formatted input
setbuf(3S): setbuf, setvbuf ..... assign buffering to a stream
stdio(3S) ..... standard buffered input/output package
system(3S) ..... issue a shell command
tmpfile(3S) ..... create a temporary file
tmpnam(3S): tmpnam, tempnam ..... create a name for a temporary file
ungetc(3S) ..... push character back into input stream
vprintf(3S): vprintf, vfprintf,
vsprintf ..... print formatted output of a varargs argument list

3SEQ Sequent I/O Library Routines

dis386(3SEQ) ..... disassemble Series 386 instructions
getgeombyname(3SEQ) ..... get disk geometry by name
getmntent(3SEQ): addmntent,
endmntent, getmntent,
hasmntopt, setmntent ..... get file system descriptor file entry
getpseudotty(3SEQ) ..... get pseudo terminals
getscsiinfo(3SEQ): getscsiinfo,
getscsimatch ..... get scsiinfo structure
getusclk(3SEQ): getusclk,
usclk_init ..... get/init microsecond clock
getzdinfo(3SEQ): getzdinfo,
getzdinfobyname ..... get zdinfo structure
initgroups(3SEQ) ..... initialize group access list
nblocks(3SEQ) ..... calculate number of blocks and indirect blocks
offline_all(3SEQ) ..... take all but one active processor off line
read_constab(3SEQ) ..... read constab entries into a structure
valloc(3SEQ) ..... aligned memory allocator

3X Special Routines

assert(3X) ..... verify program assertion
crypt(3X) ..... password and file encryption functions
curses(3X) ..... terminal screen handling and optimization package
directory(3X): opendir, readdir,
telldir, seekdir, rewinddir, closedir .... directory operations
getspent(3X): getspent, getspnam,
setspent, endspent, fgetspent,
lckpwdf, ulckpwdf ..... get shadow password file entry
ldahread(3X) ..... read the archive header of a member of an archive file
ldclose(3X): ldclose, ldaclose ..... close a common object file
ldfhread(3X) ..... read the file header of a common object file
ldgetname(3X) ..... retrieve symbol name for common object file symbol table entry

ldlread(3X): ldlread, ldlinit, ldlitem .. manipulate line number entries of a common object file function
ldlseek(3X): ldlseek, ldnlseek ..... seek to line number entries of a section of a common object file
ldohseek(3X) ..... seek to the optional file header of a common object file
ldopen(3X): ldopen, ldaopen ..... open a common object file for reading
ldrseek(3X): ldrseek, ldnrseek ..... seek to relocation entries of a section of a common object file
ldshread(3X): ldshread, ldnshread .... read an indexednamed section header of a common object file
ldsseek(3X): ldsseek, ldnsseek ..... seek to an indexednamed section of a common object file
ldtbindex(3X) ..... compute the index of a symbol table entry of a common object file
ldtbread(3X) ..... read an indexed symbol table entry of a common object file
ldtbseek(3X) ..... seek to the symbol table of a common object file
logname(3X) ..... return login name of user
malloc(3X): malloc, free, realloc, calloc, mallopt, mallinfo ..... fast main memory allocator
ndbm(3X): dbm_open, dbm_close, dbm_fetch, dbm_store, dbm_delete, dbm_firstkey, dbm_nextkey, dbm_error, dbm_clearerr, dmb_blkflush ..... database subroutines
plot(3X) ..... graphics interface subroutines
putspent(3X) ..... write shadow password file entry
regcmp(3X): regcmp, regex ..... compile and execute regular expression
sputl(3X): sputl, sgetl ..... access long integer data in a machine-independent fashion
utadduser(3X): ut_add_user, ut_delete_user ..... add/delete users in utmp file

4 File Formats

intro(4) ..... introduction to file formats
a.out(4) ..... common assembler and link editor output
acct(4) ..... per-process accounting file format
addusr_def(4) ..... add user defaults
alias(4) ..... alias file for FACE
ar(4) ..... see *C and Language Tools Binder*
backup_codes(4) ..... backup cycle code definitions
backup_codes1(4) ..... backup subcode definitions
backup_dev(4) ..... backup devices
backup_info(4) ..... backup set definitions
cftime(4) ..... language specific strings
checklist(4) ..... list of filesystems processed by fsck and ncheck
constab(4) ..... constructed devices table
core(4) ..... format of core image file
cpio(4) ..... format of cpio archive
cronlog(4) ..... cron history file
crontabs(4) ..... clock daemon file
devtab(4) ..... virtual device table

dir(4) ..... format of directories
dirent(4) ..... file system independent directory entry
diskinfo(4) ..... disk device characteristic database
dump(4): dump, dumpdates ..... incremental dump format
env(4): .environ, .pref, .variables ..... FACE files
errorlog(4) ..... internal error log file
errormsg(4) ..... error message text file
filehdr(4) ..... file header for common object files
fs(4): fs, inode ..... format of file system volume
fspec(4) ..... format specification in text files
fstab(4) ..... file-system-table
gettydefs(4) ..... speed and terminal settings used by getty
gps(4): gps ..... graphical primitive string, format of graphical files
group(4) ..... group file
helplog(4) ..... monitor help actions
init.d(4) ..... system level transition files directory
inittab(4) ..... script for the init process
issue(4) ..... issue identification file
ldfcn(4) ..... common object file access routines
limits(4) ..... file header for implementation-specific constants
linenum(4) ..... line number entries in a common object file
loginlog(4) ..... log of failed login attempts
lptypes(4) ..... table of printer types supported by the menu system
mnttab(4) ..... mounted filesystem table
mod_config(4) ..... format of a module configuration file
motd(4) ..... message-of-the-day file
ott(4) ..... files that hold object architecture information
passwd(4) ..... password file
plot(4) ..... graphics interface
product(4) ..... installed products file
profile(4) ..... setting up an environment at login time
proto(4) ..... prototype job file for at
rc(4) ..... reboot and shutdown command scripts
rcsfile(4) ..... format of RCS file
reloc(4) ..... relocation information for a common object file
sccsfile(4) ..... format of SCCS file
scnhdr(4) ..... section header for a common object file
scr_dump(4) ..... format of curses screen image file
shadow(4) ..... restricted password file
shells(4) ..... shell table
shutdown(4) ..... system shutdown information file
spellhist(4) ..... unmatched spelling words file
sulog(4) ..... log of substitute user events
syms(4) ..... common object file symbol table format
sys_config(4) ..... format of a system configuration file
syskeymap(4) ..... identify keys to internal menu system functions
task(4) ..... global security table
term(4) ..... format of compiled term file
termcap(4) ..... terminal capability data base
terminfo(4) ..... terminal capability database

timezone(4) ..... set default system time zone
unistd(4) ..... file header for symbolic constants
utmp(4): utmp, wtmp ..... utmp and wtmp entry formats
uucp(4) ..... networking support files

5 Miscellaneous

intro(5) ..... introduction to miscellany
ascii(5) ..... map of ASCII character set
environ(5) ..... user environment
fcntl(5) ..... file control options
math(5) ..... math functions and constants
prof(5) ..... profile within a function
regexp(5) ..... regular expression compile and match routines
stat(5) ..... data returned by stat system call
term(5) ..... conventional names for terminals
types(5) ..... primitive system data types
values(5) ..... machine-dependent values
varargs(5) ..... handle variable argument list

7 Special Files

intro(7) ..... introduction to special files
clone(7) ..... open any minor device on a STREAMS driver
cmpt(7) ..... compatibility partitioning driver
cn(7) ..... pseudoconsole driver
consmem(7): consmem, sm, ss ..... console memory drivers
console(7) ..... console interface
kmmem(7): kMBmem, kMWmem ..... MULITBUS memory
ld0(7) ..... STREAMS tty line discipline
log(7) ..... interface to STREAMS error logging and event tracing
lp(7) ..... Systech parallel line printer interface
lpmod(7) ..... parallel printer canonical processing module
mbad(7) ..... MULITBUS adapter
mem(7): mem, kmem ..... core memory
mtio(7) ..... UNIX magnetic tape interface
null(7) ..... the null file
pmap(7) ..... map physical addresses into processs virtual address space
sa(7) ..... devices administered by System Administration
sced(7) ..... SCSI/Ethernet/Diagnostics board
sd(7) ..... SCSI disk controller
sp(7) ..... STREAMS pipe device
ssm(7) ..... System Services Module
st(7) ..... Systech MTI-800/1600/1650 terminal multiplexor driver
streamio(7) ..... STREAMS ioctl commands
streampty(7) ..... STREAMS pseudo terminal driver
streamtty(7) ..... Generic STREAMS tty driver interface
termio(7) ..... general terminal interface
tg(7) ..... SCSI 1/2-inch magnetic tape interface

timod(7) ..... Transport Interface cooperating STREAMS module
tirdwr(7) ..... Transport Interface read/write interface STREAMS module
tm(7) ..... SCSI tape controller
ts(7) ..... SCSI tape controller
tty(7) ..... controlling terminal (gentty) interface
usclk(7) ..... 32-bit microsecond counter
vtoc(7) ..... VTOC partitioning driver
wd(7) ..... SSM SCSI disk controller
wp(7) ..... SSM parallel line printer interface
xs(7) ..... Xylogics 781 terminal multiplexor driver
xt(7) ..... Xylogics 472 MULITBUS magnetic tape interface
zd(7) ..... dual-channel disk controller

8 Stand-Alone Utilities

intro(8) ..... introduction to system standalone utilities
ccsformat(8) ..... standalone embedded SCSI disk formatter
boot(8) ..... run a standalone program or bring up the operating system
copy(8) ..... perform a standalone data transfer
dump(8) ..... standalone memory dump
prtvtoc(8) ..... print the VTOC of a standalone disk device
zdformat(8) ..... standalone disk formatter for dual-channel disk controller

Permuted Index

st: Systech MTI-800/1600/ 1650 terminal multiplexor driver.....st(7)
hpio: Hewlett-Packard 2645A terminal tape file archiver.....hpio(1)
of DASI 300 and 300s terminals 300, 300s: handle special functions.....300(1)
handle special functions of DASI 300 and 300s terminals.....300(1)
DASI 300 and 300s terminals 300s: handle special functions of.....300(1)
special functions of DASI 300 and 300s terminals.....300(1)
usclk: 32-bit microsecond counter.....usclk(7)
dis386: disassemble Series 386 instructions.....dis386(3SEQ)
DASI 450 terminal 450: handle special functions of the.....450(1)
handle special functions of the DASI 450 terminal.....450(1)
xt: Xylogics 472 MULITBUS magnetic tape interface.....xt(7)
between long integer and base- 64 ASCII string.....a64l(3C)
xs: Xylogics 781 terminal multiplexor driver.....xs(7)
integer and base-64 ASCII string a64l, l64a: convert between long.....a64l(3C)
abort: generate an IOT fault.....abort(3C)
abs: return integer absolute value.....abs(3C)
abs: return integer absolute value.....abs(3C)
fabs: floor, ceiling, remainder, absolute value functions.....floor(3M)
t_accept: accept a connect request.....t_accept(3N)
requests accept, reject: allow or prevent LP.....accept(1M)
utime: set file access and modification times.....utime(2)
touch: update access and modification times of a file.....touch(1)
file access: determine accessibility of a.....access(2)
initgroups: initialize group access list.....initgroups(3SEQ)
setgroups: set group access list.....setgroups(2SEQ)
machine-independent/ sputl, sgetl: access long integer data in a.....sputl(3X)
ldfcn: common object file access routines.....ldfcn(4)
endutent, utmpname, getutpid: access utmp file entry.....getut(3C)
access: determine accessibility of a file.....access(2)
acct: enable or disable process accounting.....acct(2)
acctcon1, acctcon2: connect-time accounting.....acctcon(1M)
acctprc1, acctprc2: process accounting.....acctprc(1M)
turnacct: shell procedures for accounting.....acctsh(1M)
runacct: run daily accounting.....runacct(1M)
/accton, acctwtmp: overview of accounting and miscellaneous/.....acct(1M)
of accounting and miscellaneous accounting commands.....acct(1M)
diskusg: generate disk accounting data by user ID.....diskusg(1M)
acct: per-process accounting file format.....acct(4)
acctcom: search and print process accounting files.....acctcom(1)
acctmerg: merge or add total accounting files.....acctmerg(1M)
command summary from per-process accounting records.....acctcms(1M)
fwtmp, wtmpfix: manipulate connect accounting records.....fwtmp(1M)
accounting format acct: enable or disable process.....acct(2)
per-process accounting records acct: per-process accounting file.....acct(4)
accounting files acctcms: command summary from.....acctcms(1M)
accounting acctcom: search and print process.....acctcom(1)
acctcon1, acctcon2: connect-time.....acctcon(1M)

acctcon1, acctcon2: connect-time accounting ..... acctcon(1M)
acctwtmp: overview of accounting/ acctdisk, acctdusg, accton, ... acct(1M)
of accounting and/ acctdisk, acctdusg, accton, acctwtmp: overview ..... acct(1M)
accounting files acctmerg: merge or add total ..... acctmerg(1M)
acctdisk, acctdusg, accton, acctwtmp: overview of accounting and/ ..... acct(1M)
accounting acctprc1, acctprc2: process ..... acctprc(1M)
acctprc1, acctprc2: process accounting ..... acctprc(1M)
acctdisk, acctdusg, accton, acctwtmp: overview of accounting and/ ..... acct(1M)
release indication t_rcvrel: acknowledge receipt of an orderly ..... t_rcvrel(3N)
trig: sin, cos, tan, asin, acos, atan, atan2: trigonometric/ ..... trig(3M)
helplog: monitor help actions ..... helplog(4)
sigaction: examine and change signal actions ..... sigaction(2)
killall: kill all active processes ..... killall(1M)
offline_all: take all but one active processor off line ..... offline_all(3SEQ)
mbad: MULITBUS adapter ..... mbad(7)
addcmdopt: add a command option to a menu ..... addcmdopt(1M)
menu addformopt: add a form/SELECT window option to a ..... addformopt(1M)
addmenuopt: add a menu option to a menu ..... addmenuopt(1M)
acctmerg: merge or add total accounting files ..... acctmerg(1M)
addusr.def: add user defaults ..... addusr_def(4)
putenv: change or add value to environment ..... putenv(3C)
menu addcmdopt: add a command option to a ..... addcmdopt(1M)
ut_add_user, ut_delete_user: add/delete users in utmp file ..... utadduser(3X)
option to a menu addformopt: add a form/SELECT window ... addformopt(1M)
menu addmenuopt: add a menu option to a ..... addmenuopt(1M)
hasmntopt, setmntent: get file/ addmntent, endmntent, getmntent, ... getmntent(3SEQ)
map an open file into the processs address space ..... mmap(2SEQ)
munmap: unmap part of the processs address space ..... munmap(2SEQ)
addresses into processs virtual address space ..... pmap(7)
t_bind: bind an address to a transport endpoint ..... t_bind(3N)
pmap: map physical addresses into processs virtual address space ..... pmap(7)
addusr.def: add user defaults ..... addusr_def(4)
files admin: create and administer SCCS ..... admin(1)
print service lpfilter: administer filters used with the LP ..... lpfilter(1M)
print service lpforms: administer forms used with the LP ..... lpforms(1M)
admin: create and administer SCCS files ..... admin(1)
SA: devices administered by System Administration ..... sa(7)
SA: devices administered by System Administration ..... sa(7)
sysadm: menu interface to do system administration ..... sysadm(1)
uadmin: administrative control ..... uadmin(1M)
uadmin: administrative control ..... uadmin(2)
swap: swap administrative interface ..... swap(1M)
/usr/adm/loginlog: log of failed login attempts ..... loginlog(4)
/onproc: run a command with priority aging, PFF, swapping, or processor/ ..... noage(1M)
alarm: set a process alarm clock ..... alarm(2)
alarm: set a process alarm clock ..... alarm(2)
/display application specific alarms and/or the working indicator ..... indicator(1F)
pathalias: alias file for FACE ..... alias(4)
valloc: aligned memory allocator ..... valloc(3SEQ)
t_alloc: allocate a library structure ..... t_alloc(3N)
brk, sbrk: change data segment space allocation ..... brk(2)
free, realloc, calloc: main memory allocator ..... malloc(3C)

mallopt, mallinfo: fast main memory allocator.....malloc(3X)
shrealloc, shfree: shared memory allocator.....shmalloc(3PPS)
valloc: aligned memory allocator.....valloc(3SEQ)
editor output a.out: common assembler and link .....a.out(4)
to maintenance commands and application programs.....intro(1)
the working/ indicator: display application specific alarms and/or.....indicator(1F)
lookup apropos: locate commands by keyword.....apropos(1)
ar: archive file format.....see *C and Lang Tools Binder*
and libraries ar: maintain archives.....see *C and Lang Tools Binder*
bc: arbitrary-precision arithmetic language .....bc(1)
ott: files that hold object architecture information.....ott(4)
cpio: format of cpio archive .....cpio(4)
the archive header of a member of an archive file.....ldahread(3X)
ar: common archive file format .....see *C and Lang Tools Binder*
archive file ldahread: read the archive header of a member of an .....ldahread(3X)
2645A terminal tape file archiver.....hpio(1)
tar: tape file archiver .....tar(1)
cpio: copy file archives in and out .....cpio(1)
varargs: handle variable argument list .....varargs(5)
print formatted output of a varargs argument list .....vprintf(3S)
xargs: construct argument list(s) and execute command.....xargs(1)
getopt: get option letter from argument vector.....getopt(3C)
echo: echo arguments.....echo(1)
expr: evaluate arguments as an expression .....expr(1)
message: puts its arguments on message line .....message(1F)
bc: arbitrary-precision arithmetic language .....bc(1)
asa: interpret ASA carriage control characters.....asa(1)
characters asa: interpret ASA carriage control.....asa(1)
/localtime, gmtime, asctime, cftime, ascftime, mktime, tzset: convert/.....ctime(3C)
ascii: map of ASCII character set .....ascii(5)
ascii: map of ASCII character set.....ascii(5)
between long integer and base-64 ASCII string .....a64l(3C)
ctime, localtime, gmtime, asctime, cftime, ascftime, mktime, tzset:/ .....ctime(3C)
trig: sin, cos, tan, asin, acos, atan, atan2: trigonometric/.....trig(3M)
as: common assembler.....see *C and Lang Tools Binder*
a.out: common assembler and link editor output.....a.out(4)
assert: verify program assertion.....assert(3X)
assert: verify program assertion.....assert(3X)
system commands assist: assistance using operating.....assist(1)
/program for generating/modifying ASSIST menus or command forms.....astgen(1)
assist: assistance using operating system commands .....assist(1)
generating/modifying ASSIST menus/ astgen: program for.....astgen(1)
later time at, batch: execute commands at a .....at(1)
trig: sin, cos, tan, asin, acos, atan, atan2: trigonometric functions.....trig(3M)
/sin, cos, tan, asin, acos, atan, atan2: trigonometric functions.....trig(3M)
double-precision number strtod, atof: convert string to.....strtod(3C)
strtol, atoi: convert string to integer.....strtol(3C)
strtol, atol, atoi: convert string to integer.....strtol(3C)
change login password and password attributes.....passwd(1)
control over various process attributes .....proc_ctl(2SEQ)
rcs: change RCS file attributes.....rcs(1)
dumpconf: dump kernel auto-configuration device table.....dumpconf(1M)

language awk: pattern scanning and processing.....awk(1)
backup.codes: backup cycle code definitions.....backup_codes(4)
backup.dev: backup devices.....backup_dev(4)
ckbupscd: check filesystem backup schedule .....ckbupscd(1M)
backup.info: backup set definitions.....backup_info(4)
backup.codes1: backup subcode definitions.....backup_codes1(4)
definitions backup.codes: backup cycle code .....backup_codes(4)
definitions backup.codes1: backup subcode.....backup_codes1(4)
backup.dev: backup devices .....backup_dev(4)
backup.info: backup set definitions.....backup_info(4)
banner: make posters.....banner(1)
m_sync: check in at barrier.....m_sync(3PPS)
initialize barrier, wait at barrier.....s_wait_barrier(3PPS)
/s_wait_barrier: initialize barrier, wait at barrier .....s_wait_barrier(3PPS)
of pathnames basename, dirname: deliver portions.....basename(1)
at, batch: execute commands at a later time.....at(1)
language bc: arbitrary-precision arithmetic .....bc(1)
procedures bcheckrc: system initialization.....bcheckrc(1M)
bdflush: update daemon.....bdflush(1M)
bdflush: update daemon.....bdflush(2SEQ)
bdiff: big diff.....bdiff(1)
bessel: j0, j1, jn, y0, y1, yn: Bessel functions .....bessel(3M)
Bessel functions bessel: j0, j1, jn, y0, y1, yn: .....bessel(3M)
bfs: big file scanner.....bfs(1)
fread, fwrite: binary input/output.....fread(3S)
bp: binary patcher.....bp(1M)
bsearch: binary search a sorted table.....bsearch(3C)
tfind, tdelete, twalk: manage binary search trees.....tsearch(3C)
tmp_affinity: bind a process to a processor.....tmp_affinity(2SEQ)
t_bind: bind an address to a transport endpoint.....t_bind(3N)
sum: print checksum and block count of a file.....sum(1)
prtvtoc: print the VTOC of a block device .....prtvtoc(1M)
sigprocmask: examine and change blocked signals.....sigprocmask(2)
number of blocks and indirect blocks.....nblocks(3SEQ)
nblocks: calculate number of blocks and indirect blocks.....nblocks(3SEQ)
df: report number of free disk blocks and inodes.....df(1M)
consistency check based on the BOM.....bomverify(1M)
consistency check based on the BOM bomverify: perform an installation.....bomverify(1M)
bootflags: display/change system boot flags.....bootflags(1M)
installboot: installs a boot program on a disk.....installboot(1M)
bring up the operating system boot: run a standalone program or.....boot(8)
boot flags bootflags: display/change system .....bootflags(1M)
bp: binary patcher.....bp(1M)
allocation brk, sbrk: change data segment space.....brk(2)
segment size brk, sbrk: change private data.....brk(3PPS)
table bsearch: binary search a sorted .....bsearch(3C)
stdio: standard buffered input/output package .....stdio(3S)
setbuf, setvbuf: assign buffering to a stream.....setbuf(3S)
mailbug: submit a bug report or enhancement request .....mailbug(1)
devbuild, devdestroy: build and destroy virtual devices .....devbuild(1M)
sccstorcs: build RCS file from SCCS file.....sccstorcs(1)
mknod: build special file.....mknod(1M)

config: build system configuration files.....config(1M)
swab: swap bytes.....swab(3C)
cc: C compiler.....see *C and Lang Tools Binder*
cflow: generate C flowgraph.....see *C and Lang Tools Binder*
cpp: the C language preprocessor.....see *C and Lang Tools Binder*
cb: C program beautifier.....see *C and Lang Tools Binder*
lint: a C program checker.....see *C and Lang Tools Binder*
cxref: generate a C program cross-reference.....see *C and Lang Tools Binder*
ctrace: C program debugger.....see *C and Lang Tools Binder*
list: produce C source listing.....see *C and Lang Tools Binder*
cal: print calendar.....cal(1)
indirect blocks nblocks: calculate number of blocks and.....nblocks(3SEQ)
dc: desk calculator.....dc(1)
cal: print calendar.....cal(1)
calendar: reminder service.....calendar(1)
stat: data returned by stat system call.....stat(5)
syscall: indirect system call.....syscall(2)
cu: call another UNIX system.....cu(1C)
malloc, free, realloc, calloc: main memory allocator.....malloc(3C)
malloc, free, realloc, calloc, mallopt, mallinfo: fast main memory/.....malloc(3X)
introduction to Sequent system calls.....intro(2SEQ)
intro: introduction to system calls and error numbers.....intro(2)
lp, cancel: send/cancel requests to the LP print service.....lp(1)
LP print service lp, cancel: send/cancel requests to the.....lp(1)
lpmod: parallel printer canonical processing module.....lpmod(7)
description into a terminfo/ captoinfo: convert a termcap.....captoinfo(1M)
asa: interpret ASA carriage control characters.....asa(1)
cat: concatenate and print files.....cat(1)
cb: C program beautifier.....see *C and Lang Tools Binder*
gencc: create a front-end to the cc command.....gencc(1M)
cc: C compiler.....see *C and Lang Tools Binder*
disk formatter CCSformat: standalone embedded SCSI.....ccsformat(8)
cd: change working directory.....cd(1)
an SCCS delta cdc: change the delta commentary of.....cdc(1)
remainder, absolute value/ floor, ceil, fmod, fabs: floor, ceiling,.....floor(3M)
floor, ceil, fmod, fabs: floor, ceiling, remainder, absolute value/.....floor(3M)
cflow: generate C flowgraph...see *C and Lang Tools Binder*
ctime, localtime, gmtime, asctime, cftime, ascftime, mktime, tzset:/.....ctime(3C)
cftime: language specific strings.....cftime(4)
sigprocmask: examine and change blocked signals.....sigprocmask(2)
brk, sbrk: change data segment space allocation.....brk(2)
attributes passwd: change login password and password.....passwd(1)
chmod: change mode.....chmod(1)
chmod: change mode of file.....chmod(2)
putenv: change or add value to environment.....putenv(3C)
chown: change owner and group of a file.....chown(2)
chown, chgrp: change owner or group.....chown(1)
nice: change priority of a process.....nice(2)
brk, sbrk: change private data segment size.....brk(3PPS)
rcs: change RCS file attributes.....rcs(1)
chroot: change root directory.....chroot(2)
chroot: change root directory for a command.....chroot(1M)

shbrk, shsbrk: change shared data segment size ..... shbrk(3PPS)
sigaction: examine and change signal actions ..... sigaction(2)
bootflags: display/ change system boot flags ..... bootflags(1M)
shutdown: shut down system, change system state ..... shutdown(1M)
cdc: change the delta commentary of an SCCS delta ..... cdc(1)
newform: change the format of a text file ..... newform(1)
helpadm: change the Help Facility database ..... helpadm(1M)
rename: change the name of a file ..... rename(2)
delta: make a delta (change) to an SCCS file ..... delta(1)
parameters vm_ctl: examine and change virtual memory tuning ..... vm_ctl(2SEQ)
cd: change working directory ..... cd(1)
chdir: change working directory ..... chdir(2)
pipe: create an interprocess channel ..... pipe(2)
zd: dual- channel disk controller ..... zd(7)
standalone disk formatter for dual- channel disk controller ..... zdformat(8)
strstr: find first occurrence of character ..... strstr(3C)
ungetc: push character back into input stream ..... ungetc(3S)
conversion tables chrtbl: generate character classification and ..... chrtbl(1M)
_tolower, _toupper, setchrclass: character handling ..... ctype(3C)
cuserid: get character login name of the user ..... cuserid(3S)
getc, getchar, fgetc, getw: get character or word from a stream ..... getc(3S)
putc, putchar, fputc, putw: put character or word on a stream ..... putc(3S)
ascii: map of ASCII character set ..... ascii(5)
fgrep: search a file for a character string ..... fgrep(1)
diskinfo: disk device characteristic database ..... diskinfo(4)
asa: interpret ASA carriage control characters ..... asa(1)
_tolower, toascii: translate characters ..... conv(3C)
tr: translate characters ..... tr(1)
lastlogin, monacct, nulladm,/ chargefee, ckpacct, dodisk, ... acctsh(1M)
chdir: change working directory ..... chdir(2)
filesystem directory consistency check ..... dcheck(1M)
filesystem storage consistency check ..... icheck(1M)
checkfsys: check a filesystem ..... checkfsys(1M)
fsck: filesystem consistency check and interactive repair ..... fsck(1M)
perform an installation consistency check based on the BOM ..... bomverify(1M)
checkeq: check documents for correct eqn syntax ..... checkeq(1)
ckbupscd: check filesystem backup schedule ..... ckbupscd(1M)
m_sync: check in at barrier ..... m_sync(3PPS)
ci: check in RCS revisions ..... ci(1)
co: check out RCS revisions ..... co(1)
permissions file uucheck: check the uucp directories and ..... uucheck(1M)
eqn syntax. checkeq: check documents for correct ..... checkeq(1)
checkfsys: check a filesystem ..... checkfsys(1M)
processed by fsck and ncheck checklist: list of filesystems ..... checklist(4)
sum: print checksum and block count of a file ..... sum(1)
chown, chgrp: change owner or group ..... chown(1)
/m_rele_procs: suspend and resume child process execution ..... m_park_procs(3PPS)
times: get process and child process times ..... times(2)
wait, waitpid: wait for a child process to stop or terminate ..... wait(2)
m_get_numprocs: return number of child processes ..... m_get_numprocs(3PPS)
m_kill_procs: kill child processes ..... m_kill_procs(3PPS)
m_set_procs: set number of child processes ..... m_set_procs(3PPS)

chmod: change mode ..... chmod(1)
chmod: change mode of file ..... chmod(2)
file chown: change owner and group of a ..... chown(2)
chown, chgrp: change owner or group ..... chown(1)
chroot: change root directory ..... chroot(2)
command chroot: change root directory for a ..... chroot(1M)
classification and conversion/ chrtbl: generate character ..... chrtbl(1M)
ci: check in RCS revisions ..... ci(1)
schedule ckbupscd: check filesystem backup ..... ckbupscd(1M)
nulladm, prctmp,/ chargefee, ckpacct, dodisk, lastlogin, monacct, ... acctsh(1M)
chrtbl: generate character classification and conversion tables ..... chrtbl(1M)
clri: clear inode ..... clri(1M)
ferror, feof, clearerr, fileno: stream status inquiries ..... ferror(3S)
a shell (command interpreter) with C-like syntax ..... csh(1)
alarm: set a process alarm clock ..... alarm(2)
usclk_init: get/init microsecond clock ..... getusclk(3SEQ)
cron: clock daemon ..... cron(1M)
crontabs: clock daemon file ..... crontabs(4)
clock: report CPU time used ..... clock(3C)
STREAMS driver clone: open any minor device on a ..... clone(7)
ldclose, ldaclose: close a common object file ..... ldclose(3X)
close: close a file descriptor ..... close(2)
t_close: close a transport endpoint ..... t_close(3N)
close: close a file descriptor ..... close(2)
fclose, fflush: close or flush a stream ..... fclose(3S)
/telldir, seekdir, rewinddir, closedir: directory operations ..... directory(3X)
clri: clear inode ..... clri(1M)
cmp: compare two files ..... cmp(1)
driver cmpt: compatibility partitioning ..... cmpt(7)
cn: pseudoconsole driver ..... cn(7)
co: check out RCS revisions ..... co(1)
communicate to a/ cocreate, cosend, cocheck, coreceive, codestroy: ..... coproc(1F)
coreceive, codestroy: communicate/ cocreate, cosend, cocheck, ... coproc(1F)
backup.codes: backup cycle code definitions ..... backup_codes(4)
m_multi: mark single-process code section ..... m_single(3PPS)
/cosend, cocheck, coreceive, codestroy: communicate to a process ..... coproc(1F)
col: filter reverse line-feeds ..... col(1)
to form error log dmesg: collect system diagnostic messages ..... dmesg(1M)
setcolor: redefine or create a color ..... setcolor(1F)
comb: combine SCCS deltas ..... comb(1)
comb: combine SCCS deltas ..... comb(1)
to two sorted files comm: select or reject lines common ..... comm(1)
mcs: manipulate the object file comment section ..... mcs(1)
cdc: change the delta commentary of an SCCS delta ..... cdc(1)
a.out: common assembler and link editor output ..... a.out(4)
ldclose, ldaclose: close a common object file ..... ldclose(3X)
ldfhread: read the file header of a common object file ..... ldfhread(3X)
number entries of a section of a common object file ..... ldlseek(3X)
to the optional file header of a common object file ..... ldohseek(3X)
relocation entries of a section of a common object file ..... ldrseek(3X)
indexednamed section header of a common object file ..... ldshread(3X)
to an indexednamed section of a common object file ..... ldsseek(3X)

index of a symbol table entry of a common object file ..... ldtbindex(3X)
an indexed symbol table entry of a common object file ..... ldtbread(3X)
seek to the symbol table of a common object file ..... ldtbseek(3X)
linenum: line number entries in a common object file ..... linenum(4)
reloc: relocation information for a common object file ..... reloc(4)
scnhdr: section header for a common object file ..... scnhdr(4)
ldfcn: common object file access routines ..... ldfcn(4)
ldopen, ldaopen: open a common object file for reading ..... ldopen(3X)
/manipulate line number entries of a common object file function ..... ldlread(3X)
ldgetname: retrieve symbol name for common object file symbol table/ ..... ldgetname(3X)
syms: common object file symbol table format ..... syms(4)
filehdr: file header for common object files ..... filehdr(4)
symbols glossary: definitions of common operating system terms and ..... glossary(1)
comm: select or reject lines common to two sorted files ..... comm(1)
cocheck, coreceive, codestroy: communicate to a process ..... coproc(1F)
ipcs: report inter-process communication facilities status ..... ipcs(1)
stdipc: ftok: standard interprocess communication package ..... stdipc(3C)
tfipc: interprocess communication reporter ..... tfipc(1M)
descriptions infocmp: compare or print out terminfo ..... infocmp(1M)
rcsdiff: compare RCS revisions ..... rcsdiff(1)
cmp: compare two files ..... cmp(1)
sccsdiff: compare two versions of an SCCS file ..... sccsdiff(1)
diff: differential file comparison ..... diff(1)
diff3: 3-way differential file comparison ..... diff3(1)
dircmp: directory comparison ..... dircmp(1)
cmpt: compatibility partitioning driver ..... cmpt(7)
regcmp: regular expression compile ..... regcmp(1)
regcmp, regex: compile and execute regular expression ..... regcmp(3X)
regexp: regular expression compile and match routines ..... regexp(5)
term: format of compiled term file ..... term(4)
cc: C compiler ..... see *C and Lang Tools Binder*
ec: forms compiler ..... ec(1M)
tic: terminfo compiler ..... tic(1M)
yacc: yet another compiler- compiler ..... yacc(1)
yacc: yet another compiler-compiler ..... yacc(1)
erf, erfc: error function and complementary error function ..... erf(3M)
cprs: compress an object file ..... see *C and Lang Tools Binder*
compress, uncompress, zcat: compress and expand data ..... compress(1)
pack, pcat, unpack: compress and expand files ..... pack(1)
and expand data compress, uncompress, zcat: compress ..... compress(1)
entry of a common object/ ldtbindex: compute the index of a symbol table ..... ldtbindex(3X)
cat: concatenate and print files ..... cat(1)
test: condition evaluation command ..... test(1)
files config: build system configuration ..... config(1M)
pathconf, fpathconf: get configurable pathname variables ..... pathconf(2)
sysconf: get configurable system variables ..... sysconf(2)
dumpconf: dump kernel auto- configuration device table ..... dumpconf(1M)
mod_config: format of a module configuration file ..... mod_config(4)
sys_config: format of a system configuration file ..... sys_config(4)
config: build system configuration files ..... config(1M)
showcfg: show the configuration of the machine ..... showcfg(1M)
lpadmin: configure the LP print service ..... lpadmin(1M)

t_rcvconnect: receive the — confirmation from a connect request ..... t_rcvconnect(3N)
fwtmp, wtmpfix: manipulate — connect accounting records ..... fwtmp(1M)
t_accept: accept a — connect request ..... t_accept(3N)
t_listen: listen for a — connect request ..... t_listen(3N)
receive the confirmation from a — connect request ..... t_rcvconnect(3N)
establish an out-going terminal line — connection ..... dial(3C)
data or expedited data sent over a — connection ..... t_rcv(3N)
send data or expedited data over a — connection ..... t_snd(3N)
t_connect: establish a — connection with another transport user ..... t_connect(3N)
acctcon1, acctcon2: — connect-time accounting ..... acctcon(1M)
dcheck: filesystem directory — consistency check ..... dcheck(1M)
icheck: filesystem storage — consistency check ..... icheck(1M)
fsck: filesystem — consistency check and interactive repair ..... fsck(1M)
bomverify: perform an installation — consistency check based on the BOM ..... bomverify(1M)
quotacheck: filesystem quota — consistency checker ..... quotacheck(1M)
drivers — consmem, sm, ss: console memory ..... consmem(7)
— console: console interface ..... console(7)
console: — console interface ..... console(7)
consmem, sm, ss: — console memory drivers ..... consmem(7)
— constab: constructed devices table ..... constab(4)
read_constab: read — constab entries into a structure ..... read_constab(3SEQ)
header for implementation-specific — constants ..... limits(4)
math: math functions and — constants ..... math(5)
unistd: file header for symbolic — constants ..... unistd(4)
constab: — constructed devices table ..... constab(4)
deroff: remove nroff, tbl, and eqn — constructs ..... deroff(1)
Uutry: try to — contact remote system with debugging on ..... uutry(1M)
ls: list — contents of directory ..... ls(1)
sigcontext: get signal — context ..... sigcontext(2SEQ)
set and/or get signal stack — context ..... sigstack(2SEQ)
csplit: — context split ..... csplit(1)
sd: SCSI disk — controller ..... sd(7)
tm: SCSI tape — controller ..... tm(7)
ts: SCSI tape — controller ..... ts(7)
wd: SSM SCSI disk — controller ..... wd(7)
zd: dual-channel disk — controller ..... zd(7)
disk formatter for dual-channel disk — controller ..... zdformat(8)
synchronize a co-process with the — controlling FMLI object ..... vsig(1F)
virtually hangup the specified — controlling terminal ..... fvhangup(2SEQ)
tty: — controlling terminal (gentty) interface ..... tty(7)
_tolower, toascii: translate/ — conv: toupper, tolower, _toupper, ... conv(3C)
units: — conversion program ..... units(1)
character classification and — conversion tables ..... chrtbl(1M)
terminfo description captoinfo: — convert a termcap description into a ..... captoinfo(1M)
dd: — convert and copy a file ..... dd(1M)
long integers l3tol, ltol3: — convert between 3-byte integers and ..... l3tol(3C)
base-64 ASCII string a64l, l64a: — convert between long integer and ..... a64l(3C)
dev_to_ttyname, ttyname_to_dev: — convert major/minor numbers ..... dev_to_ttyname(3C)
/cftime, ascftime, mktime, tzset: — convert date and time to string ..... ctime(3C)
ecvt, fcvt, gcvt: — convert floating-point number to string ..... ecvt(3C)
scanf, fscanf, sscanf: — convert formatted input ..... scanf(3S)
strtod, atof: — convert string to double-precision number ..... strtod(3C)

strtol, atol, atoi: convert string to integer ..... strtol(3C)
password file scheme pwunconv: converts from a two to a one ..... pwunconv(1M)
vsig: synchronize a co-process with the controlling FMLI object ..... vsig(1F)
UNIX-to-UNIX system copy ..... uucp(1C)
system file copy ..... uuto(1C)
dd: convert and copy a file ..... dd(1M)
cpset: copy file and set mode, user, and group IDs ..... cpset(1)
cpio: copy file archives in and out ..... cpio(1)
cp, ln, mv: copy, link or move files ..... cp(1)
volcopy: make literal copy of filesystem ..... volcopy(1M)
transfer copy: perform a standalone data ..... copy(8)
savecore: save a core dump of the operating system ..... savecore(1M)
core: format of core image file ..... core(4)
core: format of core image file ..... core(4)
mem, kmem: core memory ..... mem(7)
and unset environment variables in core or in files ..... set(1F)
cocreate, cosend, cocheck, coreceive, codestroy: communicate to a/ ..... coproc(1F)
checkeq: check documents for correct eqn syntax ..... checkeq(1)
trigonometric functions trig: sin, cos, tan, asin, acos, atan, atan2: ..... trig(3M)
codestroy: communicate to/ cocreate, cosend, cocheck, coreceive, ... coproc(1F)
sinh, cosh, tanh: hyperbolic functions ..... sinh(3M)
wc: word count ..... wc(1)
sum: print checksum and block count of a file ..... sum(1)
m_next: increment global counter ..... m_next(3PPS)
usclk: 32-bit microsecond counter ..... usclk(7)
cp, ln, mv: copy, link or move files ..... cp(1)
cpio: format of cpio archive ..... cpio(4)
cpio: copy file archives in and out ..... cpio(1)
cpio: format of cpio archive ..... cpio(4)
cpp: C preprocessor ..... see *C and Lang Tools Binder*
cprs: compress object file ..... see *C and Lang Tools Binder*
and group IDs cpset: copy file and set mode, user, ... cpset(1)
clock: report CPU time used ..... clock(3C)
cpus_online: returns the number of CPUs on-line ..... cpus_online(3PPS)
CPUs on-line cpus_online: returns the number of ..... cpus_online(3PPS)
crash: examine system images ..... crash(1M)
an existing one creat: create a new file or rewrite ..... creat(2)
setcolor: redefine or create a color ..... setcolor(1F)
gencc: create a front-end to the cc command ..... gencc(1M)
tmpnam, tempnam: create a name for a temporary file ..... tmpnam(3S)
creat: create a new file or rewrite an existing one ..... creat(2)
fork: create a new process ..... fork(2)
ctags: create a tags file ..... ctags(1)
tmpfile: create a temporary file ..... tmpfile(3S)
pipe: create an interprocess channel ..... pipe(2)
admin: create and administer SCCS files ..... admin(1)
tasktbl: create security table in shared memory ..... tasktbl(1M)
setsid: create session and set process group ID ..... setsid(2)
pathconv: search Interpreter criteria for filename ..... pathconv(1F)
cron: clock daemon ..... cron(1M)
cronlog: cron history file ..... cronlog(4)
cronlog: cron history file ..... cronlog(4)

crontab: user — crontab file ..... crontab(1)
crontab: user crontab file ..... crontab(1)
crontabs: clock daemon file ..... crontabs(4)
cxref: generate C program — cross-reference ..... see *C and Lang Tools Binder*
pg: file perusal filter for — CRTs ..... pg(1)
crypt: encode/decode ..... crypt(1)
functions — crypt: password and file encryption ..... crypt(3X)
hashing encryption — crypt, setkey, encrypt: generate ..... crypt(3C)
with C-like syntax — csh: a shell (command interpreter) ..... csh(1)
csplit: context split ..... csplit(1)
ct: spawn getty to a remote terminal ..... ct(1C)
ctags: create a tags file ..... ctags(1)
terminal — ctermid: generate filename for ..... ctermid(3S)
cftime, ascftime, mktime, tzset:/ — ctime, localtime, gmtime, asctime, ... ctime(3C)
ctrace: C prog debugger ..... see *C and Lang Tools Binder*
cu: call another UNIX system ..... cu(1C)
scr_dump: format of — curses screen image file ..... scr_dump(4)
optimization package — curses: terminal screen handling and ..... curses(3X)
the user — cuserid: get character login name of ..... cuserid(3S)
line of a file — cut: cut out selected fields of each ..... cut(1)
cut: — cut out selected fields of each line of a file ..... cut(1)
cross-reference — cxref: generate C program ..... see *C and Lang Tools Binder*
backup.codes: backup — cycle code definitions ..... backup_codes(4)
bdflush: update — daemon ..... bdflush(1M)
bdflush: update — daemon ..... bdflush(2SEQ)
cron: clock — daemon ..... cron(1M)
strerr: STREAMS error logger — daemon ..... strerr(1M)
crontabs: clock — daemon file ..... crontabs(4)
creation pwdhd: — daemon to oversee password database ..... pwdhd(1M)
runacct: run — daily accounting ..... runacct(1M)
300s: handle special functions of — DASI 300 and 300s terminals ..... 300(1)
450: handle special functions of the — DASI 450 terminal ..... 450(1)
zcat: compress and expand — data ..... compress(1)
prof: display profile — data ..... prof(1)
time a command; report process — data and system activity ..... timex(1)
termcap: terminal capability — data base ..... termcap(4)
diskusg: generate disk accounting — data by user ID ..... diskusg(1M)
t_rcvuderr: receive a unit — data error indication ..... t_rcvuderr(3N)
sputl, sgetl: access long integer — data in a machine-independent/ ..... sputl(3X)
plock: lock process, text, or — data in memory ..... plock(2)
t_snd: send — data or expedited data over a connection ..... t_snd(3N)
t_rcv: receive — data or expedited data sent over a connection ..... t_rcv(3N)
t_snd: send data or expedited — data over a connection ..... t_snd(3N)
stat: — data returned by stat system call ..... stat(5)
brk, sbrk: change private — data segment size ..... brk(3PPS)
shbrk, shsbrk: change shared — data segment size ..... shbrk(3PPS)
brk, sbrk: change — data segment space allocation ..... brk(2)
t_rcv: receive data or expedited — data sent over a connection ..... t_rcv(3N)
copy: perform a standalone — data transfer ..... copy(8)
types: primitive system — data types ..... types(5)
t_rcvudata: receive a — data unit ..... t_rcvudata(3N)
t_sndudata: send a — data unit ..... t_sndudata(3N)

diskinfo: disk device characteristic database....diskinfo(4)
helpadm: change the Help Facility database....helpadm(1M)
terminfo: terminal capability database....terminfo(4)
a terminal or query terminfo database....tput(1)
pwdhd: daemon to oversee password database creation....pwdhd(1M)
mkpwdbm: make a database for fast password lookup....mkpwdbm(1M)
join: relational database operator....join(1)
dbm_clearerr, dmb_blkflush: database subroutines....ndbm(3X)
makewhatis: generate man page databases....makealiases(1M)
date: print and set the date....date(1)
ascftime, mktime, tzset: convert date and time to string....ctime(3C)
date: print and set the date....date(1)
/dbm_nextkey, dbm_error, dbm_clearerr, dmb_blkflush: database/....ndbm(3X)
dbm_delete,/ dbm_open, dbm_close, dbm_fetch, dbm_store,....ndbm(3X)
dbm_close, dbm_fetch, dbm_store, dbm_delete, dbm_firstkey,/....ndbm(3X)
/dbm_firstkey, dbm_nextkey, dbm_error, dbm_clearerr,/....ndbm(3X)
dbm_firstkey,/dbm_open, dbm_close, dbm_fetch, dbm_store, dbm_delete,....ndbm(3X)
/dbm_fetch, dbm_store, dbm_delete, dbm_firstkey, dbm_nextkey,/....ndbm(3X)
dbm_store, dbm_delete, dbm_firstkey, dbm_nextkey, dbm_error,/....ndbm(3X)
dbm_store, dbm_delete,/ dbm_open, dbm_close, dbm_fetch, ...ndbm(3X)
dbm_open, dbm_close, dbm_fetch, dbm_store, dbm_delete, dbm_firstkey,/....ndbm(3X)
pdbx, dbx: parallel debugger....see *C and Lang Tools Binder*
dc: desk calculator....dc(1)
enable/disable patrol seeks on DCC disks....sweepon(1M)
consistency check dcheck: filesystem directory....dcheck(1M)
dd: convert and copy a file....dd(1M)
ctrace: C program debugger....see *C and Lang Tools Binder*
pdbx, dbx: parallel debugger....see *C and Lang Tools Binder*
try to contact remote system with debugging on....uutry(1M)
crypt: encode/ decode....crypt(1)
timezone: set default system time zone....timezone(4)
reset the current form field to its default values....reset(1F)
addusr.def: add user defaults....addusr_def(4)
format disks or perform other defect management functions....format(1M)
gtty: set and get terminal state (defunct)....stty(2)
ut_add_user, ut_delete_user: add/ delete users in utmp file....utadduser(3X)
the delta commentary of an SCCS delta....cdc(1)
delta: make a delta (change) to an SCCS file....delta(1)
cdc: change the delta commentary of an SCCS delta....cdc(1)
rmdel: remove a delta from an SCCS file....rmdel(1)
SCCS file delta: make a delta (change) to an....delta(1)
comb: combine SCCS deltas....comb(1)
constructs deroff: remove nroff, tbl, and eqn....deroff(1)
whatis: describe what a command is....whatis(1)
close: close a file descriptor....close(2)
dup: duplicate an open file descriptor....dup(2)
dup2: duplicate an open file descriptor....dup2(2)
setmntent: get file system descriptor file entry....getmntent(3SEQ)
getdtablemax: get maximum in-use descriptor table index....getdtablemax(2SEQ)
getdtablesize: get descriptor table size....getdtablesize(2SEQ)
setdtablesize: set descriptor table size....setdtablesize(2SEQ)
dc: desk calculator....dc(1)

devbuild, devdestroy: build and destroy virtual devices — destroy virtual devices ..... devbuild(1M)
devbuild, devdestroy: build and ..... devbuild(1M)
virtual devices devbuild, devdestroy: build and destroy ..... devbuild(1M)
ioctl: control device ..... ioctl(2)
prtvtoc: print the VTOC of a block device ..... prtvtoc(1M)
print the VTOC of a standalone disk device ..... prtvtoc(8)
sp: STREAMS pipe device ..... sp(7)
or the terminal associated with a device ..... ttydevname(1)
the terminal/ ttydevname: get the device associated with a terminal or ..... ttydevname(1)
diskinfo: disk device characteristic database ..... diskinfo(4)
devnm: device name ..... devnm(1M)
clone: open any minor device on a STREAMS driver ..... clone(7)
devtab: virtual device table ..... devtab(4)
dump kernel auto-configuration device table ..... dumpconf(1M)
backup.dev: backup devices ..... backup_dev(4)
build and destroy virtual devices ..... devbuild(1M)
of virtual-to-physical mapping devices ..... pmap(1M)
SA: devices administered by System Administration ..... sa(7)
constab: constructed devices table ..... constab(4)
devnm: device name ..... devnm(1M)
devtab: virtual device table ..... devtab(4)
convert between major/minor numbers/ dev_to_ttyname, ttyname_to_dev: ..... dev_to_ttyname(3C)
blocks and inodes df: report number of free disk ..... df(1M)
dmesg: collect system diagnostic messages to form error log ..... dmesg(1M)
SCED: SCSI/Ethernet/ Diagnostics board ..... sced(7)
terminal line connection dial: establish an out-going ..... dial(3C)
bdiff: big diff ..... bdiff(1)
diff: differential file comparison ..... diff(1)
comparison diff3: 3-way differential file ..... diff3(1)
sdiff: side-by-side difference program ..... sdiff(1)
diff: differential file comparison ..... diff(1)
diff3: 3-way differential file comparison ..... diff3(1)
dir: format of directories ..... dir(4)
dircmp: directory comparison ..... dircmp(1)
dir: format of directories ..... dir(4)
unlink: link and unlink files and directories ..... link(1M)
mkdir: make directories ..... mkdir(1)
rm, rmdir: remove files or directories ..... rm(1)
uucheck: check the uucp directories and permissions file ..... uucheck(1M)
cd: change working directory ..... cd(1)
chdir: change working directory ..... chdir(2)
chroot: change root directory ..... chroot(2)
get pathname of current working directory ..... getcwd(3C)
system level transition files directory ..... init.d(4)
ls: list contents of directory ..... ls(1)
mkdir: make a directory ..... mkdir(2)
mvdir: move a directory ..... mvdir(1M)
rmdir: remove a directory ..... rmdir(2)
uucleanup: uucp spool directory clean-up ..... uucleanup(1M)
dircmp: directory comparison ..... dircmp(1)
dcheck: filesystem directory consistency check ..... dcheck(1M)
system independent/ getdents: read directory entries and put in a file ..... getdents(2)

dirent: file system independent — directory entry ..... dirent(4)
unlink: remove — directory entry ..... unlink(2)
chroot: change root — directory for a command ..... chroot(1M)
pwd: working — directory name ..... pwd(1)
telldir, seekdir, rewinddir,/ — directory: opendir, readdir, ... directory(3X)
seekdir, rewinddir, closedir: — directory operations ..... directory(3X)
mknod: make a — directory, or a special or ordinary file ..... mknod(2)
directory entry — dirent: file system independent ..... dirent(4)
basename, — dirname: deliver portions of pathnames ..... basename(1)
— dis: code disassembler ..... see *C and Lang Tools Binder*
instructions — dis386: disassemble Series 386 ..... dis386(3SEQ)
t_unbind: — disable a transport endpoint ..... t_unbind(3N)
enable, — disable: enable/disable LP printers ..... enable(1)
enable, disable: enable/ — disable LP printers ..... enable(1)
sweepon, sweepoff: enable/ — disable patrol seeks on DCC disks ..... sweepon(1M)
acct: enable or — disable process accounting ..... acct(2)
swapping, or processor migration — disabled ..... noage(1M)
dis386: — disassemble Series 386 instructions ..... dis386(3SEQ)
dis: object code — disassembler ..... see *C and Lang Tools Binder*
t_rcvdis: retrieve information from — disconnect ..... t_rcvdis(3N)
t_snddis: send user-initiated — disconnect request ..... t_snddis(3N)
installs a boot program on a — disk ..... installboot(1M)
diskusg: generate — disk accounting data by user ID ..... diskusg(1M)
df: report number of free — disk blocks and inodes ..... df(1M)
sd: SCSI — disk controller ..... sd(7)
wd: SSM SCSI — disk controller ..... wd(7)
zd: dual-channel — disk controller ..... zd(7)
disk formatter for dual-channel — disk controller ..... zdformat(8)
print the VTOC of a standalone — disk device ..... prtvtoc(8)
diskinfo: — disk device characteristic database ..... diskinfo(4)
CCSformat: standalone embedded SCSI — disk formatter ..... ccsformat(8)
controller zdformat: standalone — disk formatter for dual-channel disk ..... zdformat(8)
getgeombyname: get — disk geometry by name ..... getgeombyname(3SEQ)
quotactl: manipulate — disk quotas ..... quotactl(2SEQ)
du: summarize — disk usage ..... du(1M)
quota: display — disk usage and limits ..... quota(1)
database — diskinfo: disk device characteristic ..... diskinfo(4)
mkvtoc: populate VTOC on — disks ..... mkvtoc(1M)
enable/disable patrol seeks on DCC — disks ..... sweepon(1M)
mgmt functions format: format — disks or perform other defect ..... format(1M)
data by user ID — diskusg: generate disk accounting ..... diskusg(1M)
and/or the working/ indicator: — display application specific alarms ..... indicator(1F)
quota: — display disk usage and limits ..... quota(1)
vi: screen-oriented (visual) — display editor based on ex ..... vi(1)
man: — display on-line man pages ..... man(1)
virtual-to-physical mapping/ pmap: — display or alter parameters of ..... pmap(1M)
prof: — display profile data ..... prof(1)
bootflags: — display/change system boot flags ..... bootflags(1M)
hypot: Euclidean — distance function ..... hypot(3M)
— dltsectbl: refresh security table ..... dltsectbl(1M)
/dbm_error, dbm_clearerr, — dmb_blkflush: database subroutines ..... ndbm(3X)
messages to form error log — dmesg: collect system diagnostic ..... dmesg(1M)

checkeq: check documents for correct eqn syntax.....checkeq(1)
prctmp,/ chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm,.....acctsh(1M)
strtod, atof: convert string to double-precision number .....strtod(3C)
mrand48, jrand48, srand48, seed48,/ drand48, erand48, lrand48, nrand48,.....drand48(3C)
open any minor device on a STREAMS driver.....clone(7)
cmpt: compatibility partitioning driver.....cmpt(7)
cn: pseudoconsole driver.....cn(7)
terminal multiplexor driver .....st(7)
streampty: STREAMS pseudo terminal driver.....streampty(7)
vtoc: VTOC partitioning driver .....vtoc(7)
Xylogics 781 terminal multiplexor driver.....xs(7)
streamtty: Generic STREAMS tty driver interface.....streamtty(7)
consmem, sm, ss: console memory drivers.....consmem(7)
du: summarize disk usage.....du(1M)
zd: dual-channel disk controller.....zd(7)
/standalone disk formatter for dual-channel disk controller.....zdformat(8)
dump: incremental filesystem dump.....dump(1M)
dump: standalone memory dump .....dump(8)
od: octal dump.....od(1)
of an object file dump: dump parts.....see *C and Lang Tools Binder*
format dump, dumpdates: incremental dump.....dump(4)
dumpfs: dump filesystem information .....dumpfs(1M)
dump, dumpdates: incremental dump format.....dump(4)
dump: incremental filesystem dump.....dump(1M)
device table dumpconf: dump kernel auto-configuration .....dumpconf(1M)
savecore: save a core dump of the operating system.....savecore(1M)
dump: standalone memory dump.....dump(8)
auto-configuration device table dumpconf: dump kernel.....dumpconf(1M)
dump, dumpdates: incremental dump format.....dump(4)
dumpfs: dump filesystem information.....dumpfs(1M)
descriptor dup: duplicate an open file .....dup(2)
descriptor dup2: duplicate an open file .....dup2(2)
dup: duplicate an open file descriptor .....dup(2)
dup2: duplicate an open file descriptor .....dup2(2)
ec: forms compiler.....ec(1M)
echo: echo arguments.....echo(1)
echo: echo arguments .....echo(1)
echo: put string on virtual output.....echo(1F)
floating-point number to string ecvt, fcvt, gcvt: convert.....ecvt(3C)
ed, red: text editor .....ed(1)
end, etext, edata: last locations in program .....end(3C)
command edc: run an externally described.....edc(1M)
casual users) edit: text editor (variant of ex for.....edit(1)
edquota: edit user quotas .....edquota(1M)
sact: print current SCCS file editing activity.....sact(1)
ed, red: text editor.....ed(1)
ex: text editor.....ex(1)
sed: stream editor.....sed(1)
vi: screen-oriented (visual) display editor based on ex.....vi(1)
ld: link editor for object files.....see *C and Lang Tools Binder*
a.out: common assembler and link editor output.....a.out(4)
edit: text editor (variant of ex for casual users).....edit(1)

edquota: edit user quotas ..... edquota(1M)
effective user, real group, and effective group IDs ..... getuid(2)
/getgid, getegid: get real user, effective user, real group, and/ ..... getuid(2)
efl: Extended Fortran Language ..... efl(1)
fsplit: split f77, ratfor, or efl files ..... fsplit(1)
using full regular expressions egrep: search a file for a pattern ..... egrep(1)
CCSformat: standalone embedded SCSI disk formatter ..... ccsformat(8)
printers enable, disable: enable/disable LP ..... enable(1)
acct: enable or disable process accounting ..... acct(2)
enable, disable: enable/disable LP printers ..... enable(1)
sweepon, sweepoff: enable/disable patrol seeks on DCC disks ..... sweepon(1M)
crypt: encode/decode ..... crypt(1)
crypt, setkey, encrypt: generate hashing encryption ..... crypt(3C)
setkey, encrypt: generate hashing encryption ..... crypt(3C)
crypt: password and file encryption functions ..... crypt(3X)
makekey: generate encryption key ..... makekey(1)
program end, etext, edata: last locations in ..... end(3C)
gencc: create a front- end to the cc command ..... gencc(1M)
/getgrgid, getgrnam, setgrent, endgrent, fgetgrent: get group file entry ..... getgrent(3C)
setmntent: get file/ addmntent, endmntent, getmntent, hasmntopt, ... getmntent(3SEQ)
bind an address to a transport endpoint ..... t_bind(3N)
t_close: close a transport endpoint ..... t_close(3N)
at the current event on a transport endpoint ..... t_look(3N)
t_open: establish a transport endpoint ..... t_open(3N)
manage options for a transport endpoint ..... t_optmgmt(3N)
t_unbind: disable a transport endpoint ..... t_unbind(3N)
/getpwuid, getpwnam, setpwent, endpwent, fgetpwent: get password file/ ..... getpwent(3C)
getspent, getspnam, setspent, endspent, fgetspent, lckpwdf,/ ..... getspent(3X)
/getutline, pututline, setutent, endutent, utmpname, getutpid: access/ ..... getut(3C)
mailbug: submit a bug report or enhancement request ..... mailbug(1)
execution env: set environment for command ..... env(1)
files .environ, .pref, .variables: FACE ..... env(4)
environ: user environment ..... environ(5)
environ: user environment ..... environ(5)
putenv: change or add value to environment ..... putenv(3C)
commands performed for multi-user environment ..... rc2(1M)
profile: setting up an environment at login time ..... profile(4)
fpsetsticky: IEEE floating-point environment control ..... fpgetround(3C)
env: set environment for command execution ..... env(1)
getenv: return value for environment name ..... getenv(3C)
set, unset: set and unset environment variables in core or in files ..... set(1F)
deroff: remove nroff, tbl, and eqn constructs ..... deroff(1)
checkeq: check documents for correct eqn syntax ..... checkeq(1)
jrand48, srand48, seed48,/ drand48, erand48, lrand48, nrand48, mrand48, ... drand48(3C)
complementary error function erf, erfc: error function and ..... erf(3M)
complementary error function erf, erfc: error function and ..... erf(3M)
error messages perror, errno, sys_errlist, sys_nerr: system ..... perror(3C)
error function and complementary error function ..... erf(3M)
error function erf, erfc: error function and complementary ..... erf(3M)
t_rcvuderr: receive a unit data error indication ..... t_rcvuderr(3N)
system diagnostic messages to form error log ..... dmesg(1M)
errorlog: internal error log file ..... errorlog(4)

strclean: STREAMS error logger cleanup program ..... strclean(1M)
strerr: STREAMS error logger daemon ..... strerr(1M)
log: interface to STREAMS error logging and event tracing ..... log(7)
t_error: produce error message ..... t_error(3N)
errormsg: error message text file ..... errormsg(4)
errno, sys_errlist, sys_nerr: system error messages ..... perror(3C)
introduction to system calls and error numbers ..... intro(2)
matherr: error-handling function ..... matherr(3M)
errorlog: internal error log file ..... errorlog(4)
errormsg: error message text file ..... errormsg(4)
spellin, hashcheck: find spelling errors ..... spell(1)
/etc/shadow with information from / etc/passwd ..... pwconv(1M)
pwconv: installs and updates / etc/shadow with information from/ ..... pwconv(1M)
end, etext, edata: last locations in program ..... end(3C)
SCED: SCSI/ Ethernet/Diagnostics board ..... sced(7)
hypot: Euclidean distance function ..... hypot(3M)
expr: evaluate arguments as an expression ..... expr(1)
test: condition evaluation command ..... test(1)
t_look: look at the current event on a transport endpoint ..... t_look(3N)
to STREAMS error logging and event tracing ..... log(7)
sulog: log of substitute user events ..... sulog(4)
(visual) display editor based on ex ..... vi(1)
edit: text editor (variant of ex for casual users) ..... edit(1)
ex: text editor ..... ex(1)
execlp, execvp: execute a file exec: execl, execv, execle, execve, ... exec(2)
execlp, execvp: execute a/ exec: execl, execv, execle, execve, ... exec(2)
exec: execl, execv, execle, execve, execlp, execvp: execute a file ..... exec(2)
exec: execl, execv, execle, execve, execlp, execvp: execute a file ..... exec(2)
run: run an executable ..... run(1F)
execle, execve, execlp, execvp: execute a file ..... exec(2)
m_fork: execute a subprogram in parallel ..... m_fork(3PPS)
construct argument list(s) and execute command ..... xargs(1)
at, batch: execute commands at a later time ..... at(1)
regcmp, regex: compile and execute regular expression ..... regcmp(3X)
uuxqt: execute remote command requests ..... uuxqt(1M)
env: set environment for command execution ..... env(1)
suspend and resume child process execution ..... m_park_procs(3PPS)
/system command execution ..... uux(1C)
sleep: suspend execution for an interval ..... sleep(1)
sleep: suspend execution for interval ..... sleep(3C)
monitor: prepare execution profile ..... monitor(3C)
profil: execution time profile ..... profil(2)
execvp: execute a file exec: execl, execv, execle, execve, execlp, ... exec(2)
exec: execl, execv, execle, execve, execlp, execvp: execute a file ..... exec(2)
execv, execle, execve, execlp, execvp: execute a file ..... exec(2)
exit, _exit: terminate process ..... exit(2)
exit, _exit: terminate process ..... exit(2)
exponential, logarithm, power,/ exp, log, log10, pow, sqrt: ..... exp(3M)
uncompress, zcat: compress and expand data ..... compress(1)
pack, pcat, unpack: compress and expand files ..... pack(1)
exp, log, log10, pow, sqrt: exponential, logarithm, power, square/ ..... exp(3M)
expression expr: evaluate arguments as an ..... expr(1)

expr: evaluate arguments as an expression ..... expr(1)
regex: compile and execute regular expression ..... regcmp(3X)
regcmp: regular expression compile ..... regcmp(1)
regexp: regular expression compile and match routines ..... regexp(5)
for a pattern using full regular expressions ..... egrep(1)
efl: Extended Fortran Language ..... efl(1)
edc: run an externally described command ..... edc(1M)
fsplit: split f77, ratfor, or efl files ..... fsplit(1)
floor, ceil, fmod, fabs: floor, ceiling, remainder, absolute value/ ..... floor(3M)
pathalias: alias file for FACE ..... alias(4)
.environ, .pref, .variables: FACE files ..... env(4)
report inter-process communication facilities status ..... ipcs(1)
a number factor: obtain the prime factors of ..... factor(1)
factor: obtain the prime factors of a number ..... factor(1)
true, false: provide truth values ..... true(1)
abort: generate an IOT fault ..... abort(3C)
stream fclose, fflush: close or flush a ..... fclose(3S)
fcntl: file control ..... fcntl(2)
fcntl: file control options ..... fcntl(5)
number to string ecvt, fcvt, gcvt: convert floating-point ..... ecvt(3C)
fopen, freopen, fdopen: open a stream ..... fopen(3S)
ferror, feof, clearerr, fileno: stream status inquiries ..... ferror(3S)
stream status inquiries ferror, feof, clearerr, fileno: ..... ferror(3S)
for a filesystem ff: list filenames and statistics ..... ff(1M)
fclose, fflush: close or flush a stream ..... fclose(3S)
from a stream getc, getchar, fgetc, getw: get character or word ..... getc(3S)
/getgrnam, setgrent, endgrent, fgetgrent: get group file entry ..... getgrent(3C)
/getpwnam, setpwent, endpwent, fgetpwent: get password file entry ..... getpwent(3C)
gets, fgets: get a string from a stream ..... gets(3S)
/getspnam, setspent, endspent, fgetspent, lckpwdf, ulckpwdf: get/ ..... getspent(3X)
string fgrep: search a file for a character ..... fgrep(1)
cut: cut out selected fields of each line of a file ..... cut(1)
mkfifo: make a FIFO special file ..... mkfifo(3C)
access: determine accessibility of a file ..... access(2)
chmod: change mode of file ..... chmod(2)
chown: change owner and group of a file ..... chown(2)
core: format of core image file ..... core(4)
cronlog: cron history file ..... cronlog(4)
crontab: user crontab file ..... crontab(1)
crontabs: clock daemon file ..... crontabs(4)
ctags: create a tags file ..... ctags(1)
selected fields of each line of a file ..... cut(1)
dd: convert and copy a file ..... dd(1M)
make a delta (change) to an SCCS file ..... delta(1)
errorlog: internal error log file ..... errorlog(4)
errormsg: error message text file ..... errormsg(4)
execve, execlp, execvp: execute a file ..... exec(2)
get: get a version of an SCCS file ..... get(1)
group: group file ..... group(4)
issue: issue identification file ..... issue(4)
header of a member of an archive file ..... ldahread(3X)
ldaclose: close a common object file ..... ldclose(3X)

the file header of a common object file.....ldfhread(3X)
of a section of a common object file.....ldlseek(3X)
file header of a common object file.....ldohseek(3X)
of a section of a common object file.....ldrseek(3X)
section header of a common object file.....ldshread(3X)
section of a common object file.....ldsseek(3X)
table entry of a common object file.....ldtbindex(3X)
table entry of a common object file.....ldtbread(3X)
the symbol table of a common object file.....ldtbseek(3X)
number entries in a common object file.....linenum(4)
link: link to a file.....link(2)
mkfifo: make a FIFO special file.....mkfifo(3C)
mknod: build special file.....mknod(1M)
directory, or a special or ordinary file.....mknod(2)
format of a module configuration file.....mod_config(4)
motd: message-of-the-day file.....motd(4)
newform: change the format of a text file.....newform(1)
null: the null file.....null(7)
passwd: password file.....passwd(4)
files or subsequent lines of one file.....paste(1)
product: installed products file.....product(4)
prs: print an SCCS file.....prs(1)
rcsfile: format of RCS file.....rcsfile(4)
read: read from file.....read(2)
readv: read vector from file.....readv(2SEQ)
reinit: run an initialization file.....reinit(1F)
information for a common object file.....reloc(4)
rename: change the name of a file.....rename(2)
rmdel: remove a delta from an SCCS file.....rmdel(1)
compare two versions of an SCCS file.....sccsdiff(1)
sccsfile: format of SCCS file.....sccsfile(4)
sccstorcs: build RCS file from SCCS file.....sccstorcs(1)
section header for a common object file.....scnhdr(4)
format of curses screen image file.....scr_dump(4)
shadow: restricted password file.....shadow(4)
system shutdown information file.....shutdown(4)
spellhist: unmatched spelling words file.....spellhist(4)
print checksum and block count of a file.....sum(1)
symlink: make symbolic link to a file.....symlink(2SEQ)
format of a system configuration file.....sys_config(4)
tail: deliver the last part of a file.....tail(1)
term: format of compiled term file.....term(4)
tmpfile: create a temporary file.....tmpfile(3S)
create a name for a temporary file.....tmpnam(3S)
access and modification times of a file.....touch(1)
undo a previous get of an SCCS file.....unget(1)
uniq: report repeated lines in a file.....uniq(1)
add/delete users in utmp file.....utadduser(3X)
the uucp directories and permissions file.....uucheck(1M)
val: validate SCCS file.....val(1)
write: write on a file.....write(2)
writev: write vector on a file.....writev(2SEQ)

utime: set file access and modification times.....utime(2)
ldfcn: common object file access routines.....ldfcn(4)
readfile, longline: reads file and gets longest line.....readfile(1F)
cpset: copy file and set mode, user, and group IDs.....cpset(1)
Hewlett-Packard 2645A terminal tape file archiver.....hpio(1)
tar: tape file archiver.....tar(1)
cpio: copy file archives in and out.....cpio(1)
rcs: change RCS file attributes.....rcs(1)
pwck, grpck: password/group file checkers.....pwck(1M)
mcs: manipulate the object file comment section.....mcs(1)
diff: differential file comparison.....diff(1)
diff3: 3-way differential file comparison.....diff3(1)
fcntl: file control.....fcntl(2)
fcntl: file control options.....fcntl(5)
UNIX-to-UNIX system file copy.....uuto(1C)
umask: set and get file creation mask.....umask(2)
close: close a file descriptor.....close(2)
dup: duplicate an open file descriptor.....dup(2)
dup2: duplicate an open file descriptor.....dup2(2)
file: determine file type.....file(1)
sact: print current SCCS file editing activity.....sact(1)
crypt: password and file encryption functions.....crypt(3X)
endgrent, fgetgrent: get group file entry.....getgrent(3C)
get file system descriptor file entry.....getmntent(3SEQ)
endpwent, fgetpwent: get password file entry.....getpwent(3C)
ulckpwdf: get shadow password file entry.....getspent(3X)
utmpname, getutpid: access utmp file entry.....getut(3C)
putpwent: write password file entry.....putpwent(3C)
putspent: write shadow password file entry.....putspent(3X)
fgrep: search a file for a character string.....fgrep(1)
grep: search a file for a pattern.....grep(1)
egrep: search a file for a pattern using full regular expressions.....egrep(1)
proto: prototype job file for at.....proto(4)
pathalias: alias file for FACE.....alias(4)
ldaopen: open a common object file for reading.....ldopen(3X)
acct: per-process accounting file format.....acct(4)
ar: common archive file format.....see *C and Lang Tools Binder*
manager, updates text files in ISIS file format.....isismgr(1M)
intro: introduction to file formats.....intro(4)
sccstorcs: build RCS file from SCCS file.....sccstorcs(1)
number entries of a common object file function.....ldlread(3X)
filehdr: file header for common object files.....filehdr(4)
limits: file header for implementation-specific/.....limits(4)
unistd: file header for symbolic constants.....unistd(4)
ldfhread: read the file header of a common object file.....ldfhread(3X)
ldohseek: seek to the optional file header of a common object file.....ldohseek(3X)
split: split a file into pieces.....split(1)
mmap: map an open file into the processs address space.....mmap(2SEQ)
ISIS file format isismgr: ISIS file manager, updates text files in.....isismgr(1M)
merge: three-way file merge.....merge(1)
ttyslot: find the slot in the utmp file of the current user.....ttyslot(3C)
more: view a file one full screen at a time.....more(1)

fuser: identify processes using a file or file structure.....fuser(1M)
creat: create a new file or rewrite an existing one.....creat(2)
pg: file perusal filter for CRTs.....pg(1)
lseek: move read/write file pointer.....lseek(2)
fseek, rewind, ftell: reposition a file pointer in a stream.....fseek(3S)
bfs: big file scanner.....bfs(1)
from a two to a one password file scheme.....pwunconv(1M)
stat, lstat, fstat: get file status.....stat(2)
identify processes using a file or file structure.....fuser(1M)
symbol name for common object file symbol table entry.....ldgetname(3X)
syms: common object file symbol table format.....syms(4)
newfs: construct a new file system.....newfs(1M)
/read directory entries and put in a file system independent format.....getdents(2)
/getmntent, hasmntopt, setmntent: get file system descriptor file entry.....getmntent(3SEQ)
dirent: file system independent directory entry.....dirent(4)
fs, inode: format of file system volume.....fs(4)
truncate, ftruncate: truncate a file to a specified length.....truncate(2SEQ)
uusched: the scheduler for the uucp file transport program.....uusched(1M)
uucico: file transport program for the uucp system.....uucico(1M)
ftw: walk a file tree.....ftw(3C)
file: determine file type.....file(1)
umask: set file-creation mode mask.....umask(1)
object files filehdr: file header for common.....filehdr(4)
mktemp: make a unique filename.....mktemp(3C)
search Interpreter criteria for filename.....pathconv(1F)
ctermid: generate filename for terminal.....ctermid(3S)
ff: list filenames and statistics for a filesystem.....ff(1M)
ferror, feof, clearerr, fileno: stream status inquiries.....ferror(3S)
search and print process accounting files.....acctcom(1)
merge or add total accounting files.....acctmerg(1M)
admin: create and administer SCCS files.....admin(1)
cat: concatenate and print files.....cat(1)
cmp: compare two files.....cmp(1)
or reject lines common to two sorted files.....comm(1)
config: build system configuration files.....config(1M)
cp, ln, mv: copy, link or move files.....cp(1)
.environ, .pref, .variables: FACE files.....env(4)
file header for common object files.....filehdr(4)
find: find files.....find(1)
fspec: format specification in text files.....fspec(4)
fsplit: split f77, ratfor, or efl files.....fsplit(1)
string, format of graphical files.....gps(4)
ident: identify files.....ident(1)
intro: introduction to special files.....intro(7)
lockf: record locking on files.....lockf(3C)
makedev: make system special files.....makedev(1M)
pcat, unpack: compress and expand files.....pack(1)
pr: print files.....pr(1)
and other information about RCS files.....rlog(1)
environment variables in core or in files.....set(1F)
sort: sort and/or merge files.....sort(1)
uucp: networking support files.....uucp(4)

what: identify SCCS files ..... what(1)
link, unlink: link and unlink files and directories ..... link(1M)
init.d: system level transition files directory ..... init.d(4)
ISIS file manager, updates text files in ISIS file format ..... isismgr(1M)
passmgmt: password files management ..... passmgmt(1M)
rm, rmdir: remove files or directories ..... rm(1)
paste: merge same lines of several files or subsequent lines of one/ ..... paste(1)
ott: files that hold object architecture information ..... ott(4)
checkfsys: check a filesystem ..... checkfsys(1M)
list filenames and statistics for a filesystem ..... ff(1M)
mkfs: construct a filesystem ..... mkfs(1M)
mount: mount a filesystem ..... mount(2)
repquota: summarize quotas for a filesystem ..... repquota(1M)
tunefs: tune up an existing filesystem ..... tunefs(1M)
umount: unmount a filesystem ..... umount(2)
volcopy: make literal copy of filesystem ..... volcopy(1M)
ckbupscd: check filesystem backup schedule ..... ckbupscd(1M)
interactive repair fsck: filesystem consistency check and ..... fsck(1M)
dcheck: filesystem directory consistency check ..... dcheck(1M)
dump: incremental filesystem dump ..... dump(1M)
fstyp: determine filesystem identifier ..... fstyp(1M)
dumpfs: dump filesystem information ..... dumpfs(1M)
statfs, fstatfs: get filesystem information ..... statfs(2)
quotacheck: filesystem quota consistency checker ..... quotacheck(1M)
quotaon, quotaoff: turn filesystem quotas on and off ..... quotaon(1M)
restore: incremental filesystem restore ..... restore(1M)
ustat: get filesystem statistics ..... ustat(2)
fsstat: report filesystem status ..... fsstat(1M)
icheck: filesystem storage consistency check ..... icheck(1M)
mnttab: mounted filesystem table ..... mnttab(4)
labelit: provide labels for filesystems ..... labelit(1M)
mount, umount: mount and unmount filesystems ..... mount(1M)
umountall: mount, unmount multiple filesystems ..... mountall(1M)
checklist: list of filesystems processed by fsck and ncheck ..... checklist(4)
fstab: file-system-table ..... fstab(4)
greek: select terminal filter ..... greek(1)
nl: line numbering filter ..... nl(1)
pg: file perusal filter for CRTs ..... pg(1)
col: filter reverse line-feeds ..... col(1)
lpfilter: administer filters used with the LP print service ..... lpfilter(1M)
find: find files ..... find(1)
for an object library lorder: find ordering relation ..... see *C and Lang Tools Binder*
display/change system boot flags ..... bootflags(1M)
/fpgetsticky, fpsetsticky: IEEE floating-point environment control ..... fpgetround(3C)
isnan: isnand, isnanf: test for floating-point NaN (Not-A-Number) ..... isnan(3C)
ecvt, fcvt, gcvt: convert floating-point number to string ..... ecvt(3C)
ldexp, modf: manipulate parts of floating-point numbers ..... frexp(3C)
ceiling, remainder, absolute value/ floor, ceil, fmod, fabs: floor, ... floor(3M)
floor, ceil, fmod, fabs: floor, ceiling, remainder, absolute value/ ..... floor(3M)
fclose, fflush: close or flush a stream ..... fclose(3S)
a co-process with the controlling FMLI object ..... vsig(1F)
floor, ceil, fmod, fabs: floor, ceiling, remainder, absolute/ ..... floor(3M)

stream fopen, freopen, fdopen: open a ..... fopen(3S)
tcgetpgrp: get foreground process group ID..... tcgetpgrp(2)
tcsetpgrp: set foreground process group ID..... tcsetpgrp(2)
fork: create a new process..... fork(2)
system diagnostic messages to form error log..... dmesg(1M)
reset: reset the current form field to its default values ..... reset(1F)
acct: per-process accounting file format ..... acct(4)
dump, dumpdates: incremental dump format..... dump(4)
put in a file system independent format..... getdents(2)
updates text files in ISIS file format..... isismgr(1M)
common object file symbol table format..... syms(4)
management functions format: format disks or perform other defect..... format(1M)
other defect management functions format: format disks or perform ..... format(1M)
mod_config: format of a module configuration file..... mod_config(4)
sys_config: format of a system configuration file..... sys_config(4)
newform: change the format of a text file..... newform(1)
term: format of compiled term file..... term(4)
core: format of core image file..... core(4)
cpio: format of cpio archive..... cpio(4)
scr_dump: format of curses screen image file..... scr_dump(4)
dir: format of directories..... dir(4)
fs, inode: format of file system volume ..... fs(4)
gps: graphical primitive string, format of graphical files..... gps(4)
rcsfile: format of RCS file..... rcsfile(4)
sccsfile: format of SCCS file ..... sccsfile(4)
fspec: format specification in text files..... fspec(4)
strftime: format time information ..... strftime(3C)
intro: introduction to file formats..... intro(4)
utmp, wtmp: utmp and wtmp entry formats..... utmp(4)
scanf, fscanf, sscanf: convert formatted input ..... scanf(3S)
printf, fprintf, sprintf: print formatted output ..... printf(3S)
vprintf, vfprintf, vsprintf: print formatted output of a varargs/..... vprintf(3S)
pwfix: remove incorrectly formatted password entries..... pwfix(1M)
standalone embedded SCSI disk formatter..... ccsformat(8)
zdformat: standalone disk formatter for dual-channel disk/..... zdformat(8)
ASSIST menus or command forms..... astgen(1)
ec: forms compiler..... ec(1M)
lpforms: administer forms used with the LP print service..... lpforms(1M)
addformopt: add a form/SELECT window option to a menu..... addformopt(1M)
ratfor: rational Fortran dialect..... ratfor(1)
efl: Extended Fortran Language ..... efl(1)
variables pathconf, fpathconf: get configurable pathname..... pathconf(2)
fpgetround, fpsetround, fpgetmask, fpsetmask, fpgetsticky,/..... fpgetround(3C)
fpsetmask, fpgetsticky,/ fpgetround, fpsetround, fpgetmask, ... fpgetround(3C)
/fpsetround, fpgetmask, fpsetmask, fpgetsticky, fpsetsticky: IEEE/..... fpgetround(3C)
printf, sprintf: print formatted output..... printf(3S)
fpgetround, fpsetround, fpgetmask, fpsetmask, fpgetsticky, fpsetsticky:/..... fpgetround(3C)
fpgetsticky,/ fpgetround, fpsetround, fpgetmask, fpsetmask, ... fpgetround(3C)
/fpgetmask, fpsetmask, fpgetsticky, fpsetsticky: IEEE floating-point/..... fpgetround(3C)
putc, putchar, fputc, putw: put character or word on a stream..... putc(3S)
puts, fputs: put a string on a stream..... puts(3S)
getfrm: returns the current frame number..... getfrm(1F)

fread, fwrite: binary input/output ..... fread(3S)
t_free: free a library structure ..... t_free(3N)
df: report number of free disk blocks and inodes ..... df(1M)
malloc, free, realloc, calloc: main memory allocator..... malloc(3C)
mallinfo: fast main memory/ malloc, free, realloc, calloc, mallopt,..... malloc(3X)
fopen, freopen, fdopen: open a stream..... fopen(3S)
of floating-point numbers frexp, ldexp, modf: manipulate parts..... frexp(3C)
gencc: create a front-end to the cc command ..... gencc(1M)
volume fs, inode: format of file system..... fs(4)
scanf, fscanf, sscanf: convert formatted input..... scanf(3S)
list of filesystems processed by fsck and ncheck..... checklist(4)
and interactive repair fsck: filesystem consistency check ..... fsck(1M)
file pointer in a stream fseek, rewind, ftell: reposition a..... fseek(3S)
generation numbers fsirand: install random inode ..... fsirand(1M)
files fspec: format specification in text..... fspec(4)
files fsplit: split f77, ratfor, or efl ..... fsplit(1)
fsstat: report filesystem status..... fsstat(1M)
fstab: file-system-table ..... fstab(4)
stat, lstat, fstat: get file status ..... stat(2)
statfs, fstatfs: get filesystem information..... statfs(2)
identifier fstyp: determine filesystem..... fstyp(1M)
fseek, rewind, ftell: reposition a file pointer in a stream..... fseek(3S)
communication package stdipc: ftok: standard interprocess..... stdipc(3C)
specified length truncate, ftruncate: truncate a file to a..... truncate(2SEQ)
ftw: walk a file tree ..... ftw(3C)
function and complementary error function..... erf(3M)
gamma: log gamma function..... gamma(3M)
hypot: Euclidean distance function..... hypot(3M)
entries of a common object file function..... ldlread(3X)
matherr: error-handling function..... matherr(3M)
prof: profile within a function..... prof(5)
erf, erfc: error function and complementary error function..... erf(3M)
j0, j1, jn, y0, y1, yn: Bessel functions ..... bessel(3M)
crypt: password and file encryption functions..... crypt(3X)
logarithm, power, square root functions..... exp(3M)
ceiling, remainder, absolute value functions..... floor(3M)
or perform other defect management functions..... format(1M)
sinh, cosh, tanh: hyperbolic functions ..... sinh(3M)
sysi86: machine specific functions ..... sysi86(2)
keys to internal menu system functions..... syskeymap(4)
acos, atan, atan2: trigonometric functions ..... trig(3M)
math: math functions and constants ..... math(5)
intro: introduction to functions and libraries..... intro(3)
300, 300s: handle special functions of DASI 300 and 300s terminals..... 300(1)
hp: handle special functions of Hewlett-Packard terminals..... hp(1)
450: handle special functions of the DASI 450 terminal..... 450(1)
file or file structure fuser: identify processes using a ..... fuser(1M)
specified controlling terminal fvhangup: virtually hangup the ..... fvhangup(2SEQ)
fread, fwrite: binary input/output ..... fread(3S)
accounting records fwtmp, wtmpfix: manipulate connect..... fwtmp(1M)
gamma: log gamma function..... gamma(3M)
gamma: log gamma function..... gamma(3M)

ecvt, fcvt, gcvt: convert floating-point number to string .....ecvt(3C)
command gencc: create a front-end to the cc.....gencc(1M)
command forms astgen: program for generating/modifying ASSIST menus or.....astgen(1)
fsirand: install random inode generation numbers.....fsirand(1M)
rand, srand: simple random-number generator .....rand(3C)
streamtty: Generic STREAMS tty driver interface.....streamtty(7)
tty: controlling terminal (gentty) interface .....tty(7)
getgeombyname: get disk geometry by name .....getgeombyname(3SEQ)
character or word from a stream getc, getchar, fgetc, getw: get.....getc(3S)
or word from a stream getc, getchar, fgetc, getw: get character .....getc(3S)
working directory getcwd: get pathname of current.....getcwd(3C)
put in a file system independent/ getdents: read directory entries and.....getdents(2)
descriptor table index getdtablemax: get max in-use.....getdtablemax(2SEQ)
size getdtablesize: get descriptor table.....getdtablesize(2SEQ)
getuid, geteuid, getgid, getegid: get real user, effective user, real/.....getuid(2)
name getenv: return value for environment.....getenv(3C)
user, effective user, real/ getuid, geteuid, getgid, getegid: get real.....getuid(2)
number getfrm: returns the current frame.....getfrm(1F)
geometry by name getgeombyname: get disk/.....getgeombyname(3SEQ)
getuid, geteuid, getgid, getegid: get real user, effective user,/.....getuid(2)
setgrent, endgrent, fgetgrent: get/ getgrent, getgrgid, getgrnam,.....getgrent(3C)
endgrent, fgetgrent: get/ getgrent, getgrgid, getgrnam, setgrent,.....getgrent(3C)
fgetgrent: get/ getgrent, getgrgid, getgrnam, setgrent, endgrent, ...getgrent(3C)
IDs getgroups: get supplementary group.....getgroups(2)
currently marked menu items getitems: returns a list of the.....getitems(1F)
getlogin: get login name .....getlogin(3C)
file system/ addmntent, endmntent, getmntent, hasmntopt, setmntent: get....getmntent(3SEQ)
stream getmsg: get next message off a .....getmsg(2)
argument vector getopt: get option letter from.....getopt(3C)
getopt: parse command options.....getopt(1)
getopts, getoptcvt: parse command options.....getopts(1)
options getopts, getoptcvt: parse command.....getopts(1)
getpagesize: get system page size.....getpagesize(2SEQ)
getpass: read a password.....getpass(3C)
process group, and parent/ getpid, getpgrp, getppid: get process,.....getpid(2)
process, process group, and parent/ getpid, getpgrp, getppid: get.....getpid(2)
and parent process/ getpid, getpgrp, getppid: get process, process group,.....getpid(2)
utilization information get_process_stats: get resource....get_process_stats(2SEQ)
getpseudotty: get pseudo terminals.....getpseudotty(3SEQ)
getpw: get name from IUD .....getpw(3C)
setpwent, endpwent, fgetpwent: get/ getpwent, getpwuid, getpwnam,.....getpwent(3C)
fgetpwent: get/ getpwent, getpwuid, getpwnam, setpwent, endpwent,.....getpwent(3C)
endpwent, fgetpwent: get/ getpwent, getpwuid, getpwnam, setpwent,.....getpwent(3C)
stream gets, fgets: get a string from a.....gets(3S)
readfile, longline: reads file and gets longest line.....readfile(1F)
scsiinfo structure getscsiinfo, getscsimatch: get .....getscsiinfo(3SEQ)
getscsiinfo, getscsimatch: get scsiinfo structure.....getscsiinfo(3SEQ)
endspent, fgetspent, lckpwdf,/ getspent, getspnam, setspent,.....getspent(3X)
fgetspent, lckpwdf,/ getspent, getspnam, setspent, endspent,.....getspent(3X)
speed and terminal settings used by getty.....gettydefs(4)
speed, and line discipline getty: set terminal type, modes,.....getty(1M)
ct: spawn getty to a remote terminal.....ct(1C)

settings used by getty gettydefs: speed and terminal ..... gettydefs(4)
get real user, effective user, real/ getuid, geteuid, getgid, getegid: ..... getuid(2)
microsecond clock getusclk, usclk_init: get/init ..... getusclk(3SEQ)
pututline, setutent, endutent,/ getut: getutent, getutid, getutline, ... getut(3C)
pututline, setutent,/ getut: getutent, getutid, getutline, ... getut(3C)
getut: getutent, getutid, getutline, pututline, setutent,/ ..... getut(3C)
getut: getutent, getutid, getutline, pututline, setutent, endutent,/ ..... getut(3C)
/setutent, endutent, utmpname, getutpid: access utmp file entry ..... getut(3C)
getc, getchar, fgetc, getw: get character or word from a stream ..... getc(3S)
zdinfo structure getzdinfo, getzdinfobyname: get ..... getzdinfo(3SEQ)
getzdinfo, getzdinfobyname: get zdinfo structure ..... getzdinfo(3SEQ)
m_next: increment global counter ..... m_next(3PPS)
task: global security table ..... task(4)
operating system terms and symbols glossary: definitions of common ..... glossary(1)
mktime, tzset:/ ctime, localtime, gmtime, asctime, cftime, ascftime, ... ctime(3C)
setjmp, longjmp: non-local goto ..... setjmp(3C)
format of graphical files gps: graphical primitive string, ... gps(4)
primitive string, format of graphical files ..... gps(4)
of graphical files gps: graphical primitive string, format ..... gps(4)
plot: graphics interface ..... plot(4)
plot: graphics interface subroutines ..... plot(3X)
greek: select terminal filter ..... greek(1)
grep: search a file for a pattern ..... grep(1)
chown, chgrp: change owner or group ..... chown(1)
newgrp: log in to a new group ..... newgrp(1M)
run command in its own process group ..... setpgrp(1)
initgroups: initialize group access list ..... initgroups(3SEQ)
setgroups: set group access list ..... setgroups(2SEQ)
/get real user, effective user, real group, and effective group IDs ..... getuid(2)
/getppid: get process, process group, and parent process IDs ..... getpid(2)
group: group file ..... group(4)
pwck, grpck: password/ group file checkers ..... pwck(1M)
setgrent, endgrent, fgetgrent: get group file entry ..... getgrent(3C)
group: group file ..... group(4)
setpgrp: set process group ID ..... setpgrp(2)
create session and set process group ID ..... setsid(2)
tcgetpgrp: get foreground process group ID ..... tcgetpgrp(2)
tcsetpgrp: set foreground process group ID ..... tcsetpgrp(2)
copy file and set mode, user, and group IDs ..... cpset(1)
getgroups: get supplementary group IDs ..... getgroups(2)
user, real group, and effective group IDs ..... getuid(2)
setuid, setgid: set user and group IDs ..... setuid(2)
id: print user and group IDs and names ..... id(1M)
groups: show group memberships ..... groups(1)
chown: change owner and group of a file ..... chown(2)
send a signal to a process or a group of processes ..... kill(2)
maintain, update, and regenerate groups of programs ..... make(1)
groups: show group memberships ..... groups(1)
pwck, grpck: password/group file checkers ..... pwck(1M)
ssignal, gsignal: software signals ..... ssignal(3C)
stty, gtty: set and get terminal state (defunct) ..... stty(2)
terminal fvhangup: virtually hangup the specified controlling ..... fvhangup(2SEQ)

nohup: run a command immune to — hangups and quits.....nohup(1)
hsearch, hcreate, hdestroy: manage — hash search tables.....hsearch(3C)
spell, hashmake, spellin, — hashcheck: find spelling errors .....spell(1)
crypt, setkey, encrypt: generate — hashing encryption .....crypt(3C)
spelling errors spell, — hashmake, spellin, hashcheck: find.....spell(1)
addmntent, endmntent, getmntent, — hasmntopt, setmntent: get file/.....getmntent(3SEQ)
search tables hsearch, — hcreate, hdestroy: manage hash.....hsearch(3C)
hsearch, hcreate, — hdestroy: manage hash search tables .....hsearch(3C)
scnhdr: section — header for a common object file.....scnhdr(4)
filehdr: file — header for common object files.....filehdr(4)
limits: file — header for implementation-specific constants.....limits(4)
unistd: file — header for symbolic constants.....unistd(4)
ldfhread: read the file — header of a common object file.....ldfhread(3X)
ldohseek: seek to the optional file — header of a common object file .....ldohseek(3X)
/read an indexednamed section — header of a common object file.....ldshread(3X)
ldahread: read the archive — header of a member of an archive file.....ldahread(3X)
helplog: monitor — help actions.....helplog(4)
help: operating system — Help Facility.....help(1)
helpadm: change the — Help Facility database .....helpadm(1M)
— help: operating system Help Facility.....help(1)
database — helpadm: change the Help Facility .....helpadm(1M)
— helplog: monitor help actions.....helplog(4)
file archiver hpio: — Hewlett-Packard 2645A terminal tape .....hpio(1)
hp: handle special functions of — Hewlett-Packard terminals .....hp(1)
cronlog: cron — history file.....cronlog(4)
ott: files that — hold object architecture information.....ott(4)
uptime: show — how long system has been up.....uptime(1)
Hewlett-Packard terminals — hp: handle special functions of.....hp(1)
tape file archiver — hpio: Hewlett-Packard 2645A terminal .....hpio(1)
hash search tables — hsearch, hcreate, hdestroy: manage.....hsearch(3C)
sinh, cosh, tanh: — hyperbolic functions .....sinh(3M)
— hypot: Euclidean distance function .....hypot(3M)
u3b5, vax: get processor/ machid: — i386, ns32000, pdp11, u3b, u3b2,.....machid(1)
consistency check — icheck: filesystem storage.....icheck(1M)
— ident: identify files.....ident(1)
m_get_myid: return process — identification.....m_get_myid(3PPS)
issue: issue — identification file .....issue(4)
fstyp: determine filesystem — identifier .....fstyp(1M)
shmget: get shared memory segment — identifier.....shmget(2)
file and set mode, user, and group — IDs.....cpset(1)
getgroups: get supplementary group — IDs.....getgroups(2)
process group, and parent process — IDs.....getpid(2)
real group, and effective group — IDs.....getuid(2)
setuid, setgid: set user and group — IDs .....setuid(2)
id: print user and group — IDs and names .....id(1M)
/fpsetmask, fpgetsticky, fpsetsticky: — IEEE floating-point environment/.....fpgetround(3C)
core: format of core — image file.....core(4)
scr_dump: format of curses screen — image file .....scr_dump(4)
crash: examine system — images.....crash(1M)
limits: file header for — implementation-specific constants .....limits(4)
pwfix: remove — incorrectly formatted password entries.....pwfix(1M)
m_next: — increment global counter.....m_next(3PPS)

dump, dumpdates: incremental dump format.....dump(4)
dump: incremental filesystem dump.....dump(1M)
restore: incremental filesystem restore.....restore(1M)
get maximum in-use descriptor table index.....getdtablemax(2SEQ)
common/ ldtbindex: compute the index of a symbol table entry of a.....ldtbindex(3X)
common object/ ldtbread: read an indexed symbol table entry of a.....ldtbread(3X)
common/dshread, ldnshread: read an indexednamed section header of a.....ldshread(3X)
ldsseek, ldnsseek: seek to an indexednamed section of a common/.....ldsseek(3X)
receipt of an orderly release indication.....t_rcvrel(3N)
receive a unit data error indication.....t_rcvudern(3N)
specific alarms and/or the working indicator.....indicator(1F)
specific alarms and/or the working/ indicator: display application.....indicator(1F)
calculate number of blocks and indirect blocks.....nblocks(3SEQ)
syscall: indirect system call.....syscall(2)
terminfo descriptions infocmp: compare or print out.....infocmp(1M)
getusclk, usclk_init: get/ init microsecond clock.....getusclk(3SEQ)
inittab: script for the init process.....inittab(4)
initialization init, telinit: process control.....init(1M)
files directory init.d: system level transition.....init.d(4)
list initgroups: initialize group access.....initgroups(3SEQ)
init, telinit: process control initialization.....init(1M)
reinit: run an initialization file.....reinit(1F)
bcheckrc: system initialization procedures.....bcheckrc(1M)
tput: initialize a terminal or query terminfo database.....tput(1)
s_init_barrier, s_wait_barrier: initialize barrier, wait at barrier.....s_wait_barrier(3PPS)
initgroups: initialize group access list.....initgroups(3SEQ)
/s_lock, s_clock, s_unlock: initialize, lock, unlock locks.....s_lock(3PPS)
setup: initialize system for first user.....setup(1)
inittab: script for the init process.....inittab(4)
clri: clear inode.....clri(1M)
fs, inode: format of file system volume.....fs(4)
fsirand: install random inode generation numbers.....fsirand(1M)
number of free disk blocks and inodes.....df(1M)
fscanf, sscanf: convert formatted input.....scanf(3S)
ungetc: push character back into input stream.....ungetc(3S)
fread, fwrite: binary input/output.....fread(3S)
poll: STREAMS input/output multiplexing.....poll(2)
stdio: standard buffered input/output package.....stdio(3S)
uustat: uucp status inquiry and job control.....uustat(1C)
install: install commands.....install(1M)
install: install commands.....install(1M)
fsirand: install random inode generation numbers.....fsirand(1M)
on the BOM bomverify: perform an installation consistency check based.....bomverify(1M)
on a disk installboot: installs a boot program.....installboot(1M)
product: installed products file.....product(4)
installboot: installs a boot program on a disk.....installboot(1M)
with information from/ pwconv: installs and updates /etc/shadow.....pwconv(1M)
dis386: disassemble Series 386 instructions.....dis386(3SEQ)
mailx: interactive message processing system.....mailx(1)
filesystem consistency check and interactive repair.....fsck(1M)
console: console interface.....console(7)
lp: Systech parallel line printer interface.....lp(7)

mtio: UNIX magnetic tape interface.....mtio(7)
plot: graphics interface.....plot(4)
Generic STREAMS tty driver interface.....streamtty(7)
swap: swap administrative interface.....swap(1M)
termio: general terminal interface.....termio(7)
tg: SCSI 1/2-inch magnetic tape interface.....tg(7)
tty: controlling terminal (gentty) interface.....tty(7)
wp: SSM parallel line printer interface.....wp(7)
Xylogics 472 MULITBUS magnetic tape interface.....xt(7)
timod: Transport Interface cooperating STREAMS module.....timod(7)
STREAMS module tirdwr: Transport Interface read/write interface.....tirdwr(7)
Transport Interface read/write interface STREAMS module.....tirdwr(7)
plot: graphics interface subroutines.....plot(3X)
sysadm: menu interface to do system administration.....sysadm(1)
and event tracing log: interface to STREAMS error logging.....log(7)
errorlog: internal error log file.....errorlog(4)
syskeymap: identify keys to internal menu system functions.....syskeymap(4)
asa: interpret ASA carriage control characters.....asa(1)
pathconv: search Interpreter criteria for filename.....pathconv(1F)
csh: a shell (command interpreter) with C-like syntax.....csh(1)
pipe: create an interprocess channel.....pipe(2)
ipcs: report inter-process communication facilities status.....ipcs(1)
stdipc: ftok: standard interprocess communication package.....stdipc(3C)
tfipc: interprocess communication reporter.....tfipc(1M)
ncheck: generate pathnames from i-numbers.....ncheck(1M)
streamio: STREAMS ioctl commands.....streamio(7)
ioctl: control device.....ioctl(2)
abort: generate an IOT fault.....abort(3C)
semaphore set, or shared memory ID ipcrm: remove a message queue,.....ipcrm(1)
communication facilities status ipcs: report inter-process.....ipcs(1)
/isxdigit, islower, isupper, isalpha, isalnum, isspace, iscntrl, ispunct,/.....ctype(3C)
isdigit, isxdigit, islower, isupper, isalpha, isalnum, isspace, iscntrl,/.....ctype(3C)
/iscntrl, ispunct, isprint, isgraph, isascii, tolower, toupper, toascci,/.....ctype(3C)
ttyname, isatty: find name of a terminal.....ttyname(3C)
/isupper, isalpha, isalnum, isspace, iscntrl, ispunct, isprint, isgraph,/.....ctype(3C)
isalpha, isalnum, isspace, iscntrl,/ isdigit, isxdigit, islower, isupper,.....ctype(3C)
/isspace, iscntrl, ispunct, isprint, isgraph, isascii, tolower, toupper,/.....ctype(3C)
file manager, updates text files in ISIS file format.....isismgr(1M)
files in ISIS file format isismgr: ISIS file manager, updates text.....isismgr(1M)
text files in ISIS file format isismgr: ISIS file manager, updates.....isismgr(1M)
isdigit, isxdigit, islower, isupper, isalpha, isalnum, isspace,/.....ctype(3C)
floating-point NaN (Not-A-Number) isnan: isnand, isnanf: test for.....isnan(3C)
isnan: isnand, isnanf: test for floating-point NaN/.....isnan(3C)
(Not-A-Number) isnan: isnand, isnanf: test for floating-point NaN.....isnan(3C)
/isalnum, isspace, iscntrl, ispunct, isprint, isgraph, isascii, tolower,/.....ctype(3C)
/isalpha, isalnum, isspace, iscntrl, ispunct, isprint, isgraph, isascii,/.....ctype(3C)
/islower, isupper, isalpha, isalnum, isspace, iscntrl, ispunct, isprint,/.....ctype(3C)
isdigit, isxdigit, islower, issue identification file.....issue(4)
isalnum, isspace, iscntrl,/ isdigit, isupper, isalpha, isalnum, isspace,/.....ctype(3C)
getpw: get name from isxdigit, islower, isupper, isalpha,.....ctype(3C)
bessel: IUD.....getpw(3C)
j0, j1, jn, y0, y1, yn: Bessel functions.....bessel(3M)

bessel: j0, j1, jn, y0, y1, yn: Bessel functions ..... bessel(3M)
bessel: j0, j1, jn, y0, y1, yn: Bessel functions ..... bessel(3M)
uustat: uucp status inquiry and job control ..... uustat(1C)
proto: prototype job file for at ..... proto(4)
join: relational database operator ..... join(1)
/erand48, lrand48, nrand48, mrand48, jrand48, srand48, seed48, lcong48:/ ..... drand48(3C)
sigsetjmp, siglongjmp: non-local jumps ..... sigsetjmp(3C)
dumpconf: dump kernel auto-configuration device table ..... dumpconf(1M)
makekey: generate encryption key ..... makekey(1)
syskeymap: identify keys to internal menu system functions ..... syskeymap(4)
apropos: locate commands by keyword lookup ..... apropos(1)
system command locate: use keywords to identify an operating ..... locate(1)
killall: kill all active processes ..... killall(1M)
m_kill_procs: kill child processes ..... m_kill_procs(3PPS)
a group of processes kill: send a signal to a process or ..... kill(2)
kill: terminate a process ..... kill(1)
killall: kill all active processes ..... killall(1M)
kMBmem, kMWmem: MULITBUS memory ..... kmmem(7)
mem, kmem: core memory ..... mem(7)
kMBmem, kMWmem: MULITBUS memory ..... kmmem(7)
ksh, rksh: KornShell, a standard/restricted command and /..... ksh(1)
standard/restricted command and/ ksh, rksh: KornShell, a ..... ksh(1)
integers and long integers l3tol, ltol3: convert between 3-byte ..... l3tol(3C)
and base-64 ASCII string a64l, l64a: convert between long integer ..... a64l(3C)
filesystems labelit: provide labels for ..... labelit(1M)
labelit: provide labels for filesystems ..... labelit(1M)
awk: pattern scanning and processing language ..... awk(1)
bc: arbitrary-precision arithmetic language ..... bc(1)
efl: Extended Fortran Language ..... efl(1)
command and programming language ..... ksh(1)
pattern scanning and processing language ..... nawk(1)
command programming language ..... sh(1)
cftime: language specific strings ..... cftime(4)
chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp,/ ..... acctsh(1M)
/setspent, endspent, fgetspent, lckpwdf, ulckpwdf: get shadow/ ..... getspent(3X)
/mrand48, jrand48, srand48, seed48, lcong48: generate uniformly/ ..... drand48(3C)
files ld: link editor for object ..... see *C and Lang Tools Binder*
ld0: STREAMS tty line discipline ..... ld0(7)
ldclose, ldaclose: close a common object file ..... ldclose(3X)
a member of an archive file ldahread: read the archive header of ..... ldahread(3X)
for reading ldopen, ldaopen: open a common object file ..... ldopen(3X)
object file ldclose, ldaclose: close a common ..... ldclose(3X)
floating-point numbers frexp, ldexp, modf: manipulate parts of ..... frexp(3C)
routines ldfcn: common object file access ..... ldfcn(4)
common object file ldfhread: read the file header of a ..... ldfhread(3X)
common object file symbol table/ ldgetname: retrieve symbol name for ..... ldgetname(3X)
number entries of a common/ldlread, ldlinit, ldlitem: manipulate line ..... ldlread(3X)
entries of a/ ldlread, ldlinit, ldlitem: manipulate line number ..... ldlread(3X)
manipulate line number entries of a/ ldlread, ldlinit, ldlitem: ..... ldlread(3X)
number entries of a section of a/ ldlseek, ldlseek: seek to line ..... ldlseek(3X)
entries of a section of a/ ldlseek, ldnlseek: seek to line number ..... ldlseek(3X)
of a section of a common/ ldrseek, ldnrseek: seek to relocation entries ..... ldrseek(3X)

section header of a/ ldshread, ldnshread: read an indexednamed.....ldshread(3X)
section of a common object/ ldsseek, ldnsseek: seek to an indexednamed.....ldsseek(3X)
header of a common object file ldohseek: seek to the optional file.....ldohseek(3X)
object file for reading ldopen, ldaopen: open a common.....ldopen(3X)
relocation entries of a section of/ ldrseek, ldnrseek: seek to.....ldrseek(3X)
indexednamed section header of a/ ldshread, ldnshread: read an.....ldshread(3X)
indexednamed section of a common/ ldsseek, ldnsseek: seek to an.....ldsseek(3X)
symbol table entry of a common/ ldtbindex: compute the index of a.....ldtbindex(3X)
table entry of a common object file ldtbread: read an indexed symbol.....ldtbread(3X)
of a common object file ldtbseek: seek to the symbol table.....ldtbseek(3X)
truncate a file to a specified length.....truncate(2SEQ)
init.d: system level transition files directory.....init.d(4)
lexical tasks lex: generate programs for simple.....lex(1)
lex: generate programs for simple lexical tasks.....lex(1)
lsearch, lfind: linear search and update.....lsearch(3C)
intro: introduction to functions and libraries.....intro(3)
find ordering relation for an object library.....see *C and Lang Tools Binder*
introduction to Parallel Programming Library.....intro(3PPS)
t_sync: synchronize transport library.....t_sync(3N)
t_alloc: allocate a library structure.....t_alloc(3N)
t_free: free a library structure.....t_free(3N)
shell (command interpreter) with C-like syntax.....csh(1)
quota: display disk usage and limits.....quota(1)
ulimit: get and set user limits.....ulimit(2)
implementation-specific constants limits: file header for.....limits(4)
returns the number of CPUs on-line.....cpus_online(3PPS)
line: read one line.....line(1)
puts its arguments on message line.....message(1F)
all but one active processor off line.....offline_all(3SEQ)
reads file and gets longest line.....readfile(1F)
establish an out-going terminal line connection.....dial(3C)
set terminal type, modes, speed, and line discipline.....getty(1M)
ld0: STREAMS tty line discipline.....ld0(7)
set terminal type, modes, speed, and line discipline.....uugetty(1M)
man: display on-line man pages.....man(1)
linenum: line number entries in a common object file.....linenum(4)
/ldlinit, ldlitem: manipulate line number entries of a common object/.....ldlread(3X)
a common/ ldlseek, ldnlseek: seek to line number entries of a section of.....ldlseek(3X)
nl: line numbering filter.....nl(1)
cut: cut out selected fields of each line of a file.....cut(1)
lp: Systech parallel line printer interface.....lp(7)
wp: SSM parallel line printer interface.....wp(7)
line: read one line.....line(1)
lsearch, lfind: linear search and update.....lsearch(3C)
col: filter reverse line-feeds.....col(1)
common object file linenum: line number entries in a.....linenum(4)
readlink: read value of a symbolic link.....readlink(2SEQ)
link, unlink: link and unlink files and directories.....link(1M)
ld: link editor for object files.....see *C and Lang Tools Binder*
a.out: common assembler and link editor output.....a.out(4)
link: link to a file.....link(2)
cp, ln, mv: copy, link or move files.....cp(1)

link: link to a file ..... link(2)
symlink: make symbolic link to a file ..... symlink(2SEQ)
and directories link, unlink: link and unlink files ..... link(1M)
lint: a C program checker ..... see *C and Lang Tools Binder*
initgroups: initialize group access list ..... initgroups(3SEQ)
nlist: get entries from name list ..... nlist(3C)
setgroups: set group access list ..... setgroups(2SEQ)
varargs: handle variable argument list ..... varargs(5)
output of a varargs argument list ..... vprintf(3S)
ls: list contents of directory ..... ls(1)
ff: list filenames and statistics for a filesystem ..... ff(1M)
fsck and ncheck checklist: list of filesystems processed by ..... checklist(4)
getitems: returns a list of the currently marked menu items ..... getitems(1F)
t_listen: listen for a connect request ..... t_listen(3N)
listing list: produce C source ..... see *C and Lang Tools Binder*
xargs: construct argument list(s) and execute command ..... xargs(1)
volcopy: make literal copy of filesystem ..... volcopy(1M)
cp, ln, mv: copy, link or move files ..... cp(1)
ascftime, mktime, tzset:/ ctime, localtime, gmtime, asctime, cftime, ... ctime(3C)
apropos: locate commands by keyword lookup ..... apropos(1)
operating system command locate: use keywords to identify an ..... locate(1)
setlocale: set programs location ..... setlocale(3C)
end, etext, edata: last locations in program ..... end(3C)
plock: lock process, text, or data in memory ..... plock(2)
m_lock, m_unlock: lock, unlock locks ..... m_lock(3PPS)
s_clock, s_unlock: initialize, lock, unlock locks ..... s_lock(3PPS)
lockf: record locking on files ..... lockf(3C)
lockf: record locking on files ..... lockf(3C)
m_lock, m_unlock: lock, unlock locks ..... m_lock(3PPS)
s_unlock: initialize, lock, unlock locks ..... s_lock(3PPS)
diagnostic messages to form error log ..... dmesg(1M)
errorlog: internal error log file ..... errorlog(4)
gamma: log gamma function ..... gamma(3M)
newgrp: log in to a new group ..... newgrp(1M)
logging and event tracing log: interface to STREAMS error ..... log(7)
logarithm, power, square root/ exp, log, log10, pow, sqrt: exponential, ... exp(3M)
about RCS files rlog: print log messages and other information ..... rlog(1)
/usr/adm/loginlog: log of failed login attempts ..... loginlog(4)
sulog: log of substitute user events ..... sulog(4)
logarithm, power, square/ exp, log, log10, pow, sqrt: exponential, ... exp(3M)
log, log10, pow, sqrt: exponential, logarithm, power, square root/ ..... exp(3M)
strclean: STREAMS error logger cleanup program ..... strclean(1M)
strerr: STREAMS error logger daemon ..... strerr(1M)
log: interface to STREAMS error logging and event tracing ..... log(7)
/usr/adm/loginlog: log of failed login attempts ..... loginlog(4)
getlogin: get login name ..... getlogin(3C)
logname: get login name ..... logname(1)
cuserid: get character login name of the user ..... cuserid(3S)
logname: return login name of user ..... logname(3X)
passwd: change login password and password attributes ..... passwd(1)
tfmenush: assign menu system as user login shell ..... tfmenush(1M)
login: sign on ..... login(1)

setting up an environment at login time.....profile(4)
/usr/adm/ loginlog: log of failed login attempts.....loginlog(4)
logname: get login name.....logname(1)
logname: return login name of user.....logname(3X)
setjmp, longjmp: non-local goto.....setjmp(3C)
readfile, longline: reads file and gets longest line .....readfile(1F)
apropos: locate commands by keyword lookup.....apropos(1)
make a database for fast password lookup.....mkpwdbm(1M)
relation for an object library lorder: find ordering.....see *C and Lang Tools Binder*
the LP print service lp, cancel: send/cancel requests to.....lp(1)
cancel: send/cancel requests to the LP print service.....lp(1)
lpadmin: configure the LP print service .....lpadmin(1M)
administer filters used with the LP print service.....lpfilter(1M)
administer forms used with the LP print service.....lpforms(1M)
information about the status of the LP print service.....lpstat(1)
/lpshut, lpmove: start/stop the LP print service and move requests.....lpsched(1M)
enable, disable: enable/disable LP printers .....enable(1)
accept, reject: allow or prevent LP requests.....accept(1M)
interface lp: Systech parallel line printer.....lp(7)
service lpadmin: configure the LP print.....lpadmin(1M)
with the LP print service lpfilter: administer filters used.....lpfilter(1M)
the LP print service lpforms: administer forms used with.....lpforms(1M)
processing module lpmod: parallel printer canonical.....lpmod(7)
service and move/ lpsched, lpshut, lpmove: start/stop the LP print .....lpsched(1M)
the LP print service and move/ lpsched, lpshut, lpmove: start/stop.....lpsched(1M)
print service and move/ lpsched, lpshut, lpmove: start/stop the LP.....lpsched(1M)
status of the LP print service lpstat: print information about the.....lpstat(1)
supported by the menu system lptypes: table of printer types.....lptypes(4)
priorities lpusers: set printing queue .....lpusers(1M)
srand48, seed48,/ drand48, erand48, lrand48, nrand48, mrand48, jrand48,.....drand48(3C)
ls: list contents of directory.....ls(1)
update lsearch, lfind: linear search and.....lsearch(3C)
lseek: move read/write file pointer .....lseek(2)
stat, lstat, fstat: get file status .....stat(2)
integers and long integers l3tol, ltol3: convert between 3-byte.....l3tol(3C)
m4: macro processor .....m4(1)
u3b2, u3b5, vax: get processor type/ machid: i386, ns32000, pdp11, u3b,.....machid(1)
show the configuration of the machine .....showcfg(1M)
sysi86: machine specific functions .....sysi86(2)
values: machine-dependent values.....values(5)
sgetl: access long integer data in a machine-independent fashion.....sputl(3X)
m4: macro processor.....m4(1)
mtio: UNIX magnetic tape interface .....mtio(7)
tg: SCSI 1/2-inch magnetic tape interface.....tg(7)
xt: Xylogics 472 MULITBUS magnetic tape interface.....xt(7)
mt: magnetic tape manipulating program.....mt(1)
rmail: send mail to users or read mail .....mail(1)
read mail mail, rmail: send mail to users or.....mail(1)
mail, rmail: send mail to users or read mail.....mail(1)
enhancement request mailbug: submit a bug report or.....mailbug(1)
processing system mailx: interactive message.....mailx(1)
groups of programs make: maintain, update, and regenerate.....make(1)

programs intro: introduction to ..... maintenance commands and application ..... intro(1)
/ttyname_to_dev: convert between ..... major/minor numbers/tty names ..... dev_to_ttyname(3C)
regenerate groups of programs ..... make: maintain, update, and ..... make(1)
man page databases ..... makealiases, makewhatis: generate ..... makealiases(1M)
makedev: make system special files ..... makedev(1M)
makekey: generate encryption key ..... makekey(1)
databases makealiases, ... makewhatis: generate man page ..... makealiases(1M)
/free, realloc, calloc, mallopt, ... mallinfo: fast main memory allocator ..... malloc(3X)
memory allocator ..... malloc, free, realloc, calloc: main ..... malloc(3C)
mallopt, mallinfo: fast main memory/ ..... malloc, free, realloc, calloc, ... malloc(3X)
malloc, free, realloc, calloc, ... mallopt, mallinfo: fast main memory/ ..... malloc(3X)
man: display on-line man pages ..... man(1)
makealiases, makewhatis: generate ..... man page databases ..... makealiases(1M)
man: display on-line ..... man pages ..... man(1)
tsearch, tfind, tdelete, twalk: ..... manage binary search trees ..... tsearch(3C)
hsearch, hcreate, hdestroy: ..... manage hash search tables ..... hsearch(3C)
endpoint t_optmgmt: ..... manage options for a transport ..... t_optmgmt(3N)
passmgmt: password files ..... management ..... passmgmt(1M)
sigignore, sigpause: signal ..... management ..... sigset(2)
format disks or perform other defect ..... management functions ..... format(1M)
file format isismgr: ISIS file ..... manager, updates text files in ISIS ..... isismgr(1M)
mt: magnetic tape ..... manipulating program ..... mt(1)
parameters of virtual-to-physical ..... mapping devices ..... pmap(1M)
m_single, m_multi: ..... mark single-process code section ..... m_single(3PPS)
umask: set file-creation mode ..... mask ..... umask(1)
umask: set and get file creation ..... mask ..... umask(2)
regex: ..... match patterns against a string ..... regex(1F)
regular expression compile and ..... match routines ..... regexp(5)
math: ..... math functions and constants ..... math(5)
math: math functions and constants ..... math(5)
matherr: error-handling function ..... matherr(3M)
mbad: MULITBUS adapter ..... mbad(7)
comment section ..... mcs: manipulate the object file ..... mcs(1)
mem, kmem: core memory ..... mem(7)
read the archive header of a ..... member of an archive file ..... ldahread(3X)
groups: show group ..... memberships ..... groups(1)
memset: memory operations ..... memory: ..... memccpy, memchr, memcmp, memcpy, ... memory(3C)
memory operations ..... memory: memccpy, ... memchr, memcmp, memcpy, memset: ..... memory(3C)
operations ..... memory: memccpy, memchr, ... memcmp, memcpy, memset: memory ..... memory(3C)
memory: memccpy, memchr, memcmp, ... memcpy, memset: memory operations ..... memory(3C)
kMBmem, kMWmem: MULITBUS ..... memory ..... kmmem(7)
mem, kmem: core ..... memory ..... mem(7)
lock process, text, or data in ..... memory ..... plock(2)
create security table in shared ..... memory ..... tasktbl(1M)
malloc, free, realloc, calloc: main ..... memory allocator ..... malloc(3C)
calloc, mallopt, mallinfo: fast main ..... memory allocator ..... malloc(3X)
shmalloc, shrealloc, shfree: shared ..... memory allocator ..... shmalloc(3PPS)
valloc: aligned ..... memory allocator ..... valloc(3SEQ)
shmctl: shared ..... memory control operations ..... shmctl(2)
consmem, sm, ss: console ..... memory drivers ..... consmem(7)
dump: standalone ..... memory dump ..... dump(8)
spawn new process in a virtual ..... memory efficient way ..... vfork(2)

queue, semaphore set, or shared memory ID ..... ipcrm(1)
memcpy, memset: memory operations memory: memccpy, memchr, memcmp, ... memory(3C)
memchr, memcmp, memcpy, memset: memory operations ..... memory(3C)
shmop, shmat, shmdt: shared memory operations ..... shmop(2)
vmtune: tune system-wide virtual- memory parameters ..... vmtune(1M)
shmget: get shared memory segment identifier ..... shmget(2)
vm_ctl: examine and change virtual memory tuning parameters ..... vm_ctl(2SEQ)
memccpy, memchr, memcmp, memcpy, memset: memory operations ..... memory(3C)
addcmdopt: add a command option to a menu ..... addcmdopt(1M)
add a form/SELECT window option to a menu ..... addformopt(1M)
addmenuopt: add a menu option to a menu ..... addmenuopt(1M)
sysadm: menu interface to do system administration ..... sysadm(1)
a list of the currently marked menu items ..... getitems(1F)
menu: menu processor ..... menu(1M)
menumsg: route user message to menu message queue ..... menumsg(1M)
rmvopt: remove a menu option ..... rmvopt(1M)
addmenuopt: add a menu option to a menu ..... addmenuopt(1M)
menu: menu processor ..... menu(1M)
of printer types supported by the menu system ..... lptypes(4)
tfmenush: assign menu system as user login shell ..... tfmenush(1M)
syskeymap: identify keys to internal menu system functions ..... syskeymap(4)
message queue menumsg: route user message to menu ..... menumsg(1M)
for generating/modifying ASSIST menus or command forms ..... astgen(1)
merge: three-way file merge ..... merge(1)
sort: sort and/or merge files ..... sort(1)
acctmerg: merge or add total accounting files ..... acctmerg(1M)
rcsmerge: merge RCS revisions ..... rcsmerge(1)
subsequent lines of one file paste: merge same lines of several files or ..... paste(1)
merge: three-way file merge ..... merge(1)
mesg: permit or deny messages ..... mesg(1)
t_error: produce error message ..... t_error(3N)
tfnotify: send completion message ..... tfnotify(1M)
msgctl: message control operations ..... msgctl(2)
message: puts its arguments on message line ..... message(1F)
getmsg: get next message off a stream ..... getmsg(2)
putmsg: send a message on a stream ..... putmsg(2)
msgop: message operations ..... msgop(2)
mailx: interactive message processing system ..... mailx(1)
message line message: puts its arguments on ..... message(1F)
menumsg: route user message to menu message queue ..... menumsg(1M)
msgget: get message queue ..... msgget(2)
shared memory ID ipcrm: remove a message queue, semaphore set, or ..... ipcrm(1)
errormsg: error message text file ..... errormsg(4)
menumsg: route user message to menu message queue ..... menumsg(1M)
motd: message-of-the-day file ..... motd(4)
mesg: permit or deny messages ..... mesg(1)
sys_errlist, sys_nerr: system error messages ..... perror(3C)
strace: print STREAMS trace messages ..... strace(1M)
rlog: print log messages and other information about RCS files ..... rlog(1)
dmesg: collect system diagnostic messages to form error log ..... dmesg(1M)
parallel m_fork: execute a subprogram in ..... m_fork(3PPS)
identification m_get_myid: return process ..... m_get_myid(3PPS)

number of child processes m_get_numprocs: return ..... m_get_numprocs(3PPS)
getusclk, usclk_init: get/init microsecond clock ..... getusclk(3SEQ)
usclk: 32-bit microsecond counter ..... usclk(7)
aging, PFF, swapping, or processor migration disabled ..... noage(1M)
clone: open any minor device on a STREAMS driver ..... clone(7)
/convert between major/ minor numbers and tty names ..... dev_to_ttyname(3C)
/acctwtmp: overview of accounting and miscellaneous accounting commands ..... acct(1M)
intro: introduction to miscellany ..... intro(5)
mkdir: make a directory ..... mkdir(2)
mkdir: make directories ..... mkdir(1)
mkfifo: make a FIFO special file ..... mkfifo(3C)
mkfs: construct a filesystem ..... mkfs(1M)
m_kill_procs: kill child processes ..... m_kill_procs(3PPS)
mknod: build special file ..... mknod(1M)
special or ordinary file mknod: make a directory, or a ..... mknod(2)
password lookup mkpwdbm: make a database for fast ..... mkpwdbm(1M)
mktemp: make a unique filename ..... mktemp(3C)
/gmtime, asctime, cftime, ascftime, mktime, tzset: convert date and time/ ..... ctime(3C)
mkvtoc: populate VTOC on disks ..... mkvtoc(1M)
m_lock, m_unlock: lock, unlock locks ..... m_lock(3PPS)
processs address space mmap: map an open file into the ..... mmap(2SEQ)
section m_single, m_multi: mark single-process code ..... m_single(3PPS)
m_next: increment global counter ..... m_next(3PPS)
mnttab: mounted filesystem table ..... mnttab(4)
configuration file mod_config: format of a module ..... mod_config(4)
chmod: change mode ..... chmod(1)
umask: set file-creation mode mask ..... umask(1)
chmod: change mode of file ..... chmod(2)
cpset: copy file and set mode, user, and group IDs ..... cpset(1)
getty: set terminal type, modes, speed, and line discipline ..... getty(1M)
uugetty: set terminal type, modes, speed, and line discipline ..... uugetty(1M)
frexp, ldexp, modf: manipulate parts of floating-point/ ..... frexp(3C)
utime: set file access and modification times ..... utime(2)
touch: update access and modification times of a file ..... touch(1)
astgen: program for generating/ modifying ASSIST menus or command/ ..... astgen(1)
printer canonical processing module ..... lpmod(7)
SSM: System Services Module ..... ssm(7)
Interface cooperating STREAMS module ..... timod(7)
read/write interface STREAMS module ..... tirdwr(7)
mod_config: format of a module configuration file ..... mod_config(4)
/ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily/ ..... acctsh(1M)
helplog: monitor help actions ..... helplog(4)
monitor: monitor system activity ..... monitor(1M)
monitor: prepare execution profile ..... monitor(3C)
monitor: monitor system activity ..... monitor(1M)
a time more: view a file one full screen at ..... more(1)
motd: message-of-the-day file ..... motd(4)
mount: mount a filesystem ..... mount(2)
mount, umount: mount and unmount filesystems ..... mount(1M)
mount: mount a filesystem ..... mount(2)
setmnt: establish mount table ..... setmnt(1M)
filesystems mount, umount: mount and unmount ..... mount(1M)

mountall, umountall: mount, unmount multiple filesystems.....mountall(1M)
multiple filesystems mountall, umountall: mount, unmount.....mountall(1M)
mnttab: mounted filesystem table.....mnttab(4)
mvdir: move a directory.....mvdir(1M)
cp, ln, mv: copy, link or move files.....cp(1)
lseek: move read/write file pointer.....lseek(2)
start/stop the LP print service and move requests.....lpsched(1M)
and resume child process execution m_park_procs, m_rele_procs: susp/....m_park_procs(3PPS)
ptrace, mptrace: process trace facility.....ptrace(2)
drand48, erand48, lrand48, nrand48, mrand48, jrand48, srand48, seed48,/.....drand48(3C)
child process/ m_park_procs, m_rele_procs: suspend and resume....m_park_procs(3PPS)
processes m_set_procs: set number of child.....m_set_procs(3PPS)
msgctl: message control operations.....msgctl(2)
msgget: get message queue.....msgget(2)
msgop: message operations.....msgop(2)
single-process code section m_single, m_multi: mark.....m_single(3PPS)
m_sync: check in at barrier.....m_sync(3PPS)
program mt: magnetic tape manipulating.....mt(1)
st: Systech MTI-800/1600/1650 terminal multiplexor driver.....st(7)
mtio: UNIX magnetic tape interface.....mtio(7)
mbad: MULITBUS adapter.....mbad(7)
xt: Xylogics 472 MULITBUS magnetic tape interface.....xt(7)
kMBmem, kMWmem: MULITBUS memory.....kmmem(7)
mountall, umountall: mount, unmount multiple filesystems.....mountall(1M)
poll: STREAMS input/output multiplexing.....poll(2)
Systech MTI-800/1600/1650 terminal multiplexor driver.....st(7)
xs: Xylogics 781 terminal multiplexor driver.....xs(7)
tmp_ctl: multi-processor control.....tmp_ctl(2SEQ)
rc2: run commands performed for multi-user environment.....rc2(1M)
m_lock, m_unlock: lock, unlock locks.....m_lock(3PPS)
address space munmap: unmap part of the processs.....munmap(2SEQ)
cp, ln, mv: copy, link or move files.....cp(1)
mvdir: move a directory.....mvdir(1M)
devnm: device name.....devnm(1M)
getenv: return value for environment name.....getenv(3C)
getgeombyname: get disk geometry by name.....getgeombyname(3SEQ)
getlogin: get login name.....getlogin(3C)
logname: get login name.....logname(1)
pwd: working directory name.....pwd(1)
tmpnam, tempnam: create a name for a temporary file.....tmpnam(3S)
table/ ldgetname: retrieve symbol name for common object file symbol.....ldgetname(3X)
getpw: get name from IUD.....getpw(3C)
nlist: get entries from name list.....nlist(3C)
rename: change the name of a file.....rename(2)
ttyname, isatty: find name of a terminal.....ttyname(3C)
uname: get name of current operating system.....uname(2)
uname: print name of current system.....uname(1)
tty: get the name of the terminal.....tty(1)
cuserid: get character login name of the user.....cuserid(3S)
logname: return login name of user.....logname(3X)
/ldnshread: read an indexed named section header of a common object/.....ldshread(3X)
/ldnsseek: seek to an indexed named section of a common object file.....ldsseek(3X)

between major/minor numbers and tty names.....dev_to_ttyname(3C)
id: print user and group IDs and names.....id(1M)
term: conventional names for terminals.....term(5)
isnanf: test for floating-point NaN (Not-A-Number).....isnan(3C)
processing language nawk: pattern scanning and .....nawk(1)
and indirect blocks nblocks: calculate number of blocks.....nblocks(3SEQ)
of filesystems processed by fsck and ncheck.....checklist(4)
i-numbers ncheck: generate pathnames from .....ncheck(1M)
uucp: networking support files.....uucp(4)
file newform: change the format of a text .....newform(1)
newfs: construct a new file system .....newfs(1M)
newgrp: log in to a new group.....newgrp(1M)
news: print news items.....news(1)
news: print news items.....news(1)
getmsg: get next message off a stream.....getmsg(2)
nice: change priority of a process.....nice(2)
nice: run a command at low priority .....nice(1)
nl: line numbering filter .....nl(1)
nlist: get entries from name list.....nlist(3C)
object file nm: print name list of.....see *C and Lang Tools Binder*
command with priority aging, PFF,/ noage, nopff, noswap, onproc: run a.....noage(1M)
hangups and quits nohup: run a command immune to .....nohup(1)
setjmp, longjmp: non-local goto.....setjmp(3C)
sigsetjmp, siglongjmp: non-local jumps.....sigsetjmp(3C)
with priority aging, PFF,/ noage, nopff, noswap, onproc: run a command.....noage(1M)
priority aging, PFF,/ noage, nopff, noswap, onproc: run a command with .....noage(1M)
test for floating-point NaN (Not-A-Number).....isnan(3C)
seed48,/ drand48, erand48, lrand48, nrand48, mrand48, jrand48, srand48, ...drand48(3C)
deroff: remove nroff, tbl, and eqn constructs.....deroff(1)
vax: get processor/ machid: i386, ns32000, pdp11, u3b, u3b2, u3b5,.....machid(1)
null: the null file.....null(7)
null: the null file .....null(7)
/ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct,/ .....acctsh(1M)
obtain the prime factors of a number.....factor(1)
getfrm: returns the current frame number.....getfrm(1F)
test for floating-point NaN (Not-A- Number) .....isnan(3C)
convert string to double-precision number.....strtod(3C)
linenum: line number entries in a common object file.....linenum(4)
/ldlinit, ldlitem: manipulate line number entries of a common object/.....ldlread(3X)
ldlseek, ldnlseek: seek to line number entries of a section of a/.....ldlseek(3X)
rand, srand: simple random- number generator.....rand(3C)
nblocks: calculate number of blocks and indirect blocks.....nblocks(3SEQ)
m_get_numprocs: return number of child processes .....m_get_numprocs(3PPS)
m_set_procs: set number of child processes.....m_set_procs(3PPS)
cpus_online: returns the number of CPUs on-line .....cpus_online(3PPS)
df: report number of free disk blocks and inodes.....df(1M)
fcvt, gcvt: convert floating-point number to string .....ecvt(3C)
nl: line numbering filter.....nl(1)
uniformly distributed pseudorandom numbers.....drand48(3C)
manipulate parts of floating-point numbers .....frexp(3C)
install random inode generation numbers .....fsirand(1M)
to system calls and error numbers .....intro(2)

ncheck: generate pathnames from i- numbers.....ncheck(1M)
/convert between major/minor numbers and tty names.....dev_to_ttyname(3C)
cprs: compress an object file.....see *C and Lang Tools Binder*
strstr: find first occurrence of character.....strstr(3C)
od: octal dump.....od(1)
od: octal dump.....od(1)
offline: bring processors online and offline.....online(1M)
online, processor off line offline: bring processors online and offline.....online(1M)
offline_all: take all but one active.....offline_all(3SEQ)
online and offline.....online(1M)
online, offline: bring processors.....online(1M)
aging, PFF,/ noage, nopff, noswap, onproc: run a command with priority.....noage(1M)
ldopen, ldaopen: open a common object file for reading.....ldopen(3X)
fopen, freopen, fdopen: open a stream.....fopen(3S)
clone: open any minor device on a STREAMS driver.....clone(7)
dup: duplicate an open file descriptor.....dup(2)
dup2: duplicate an open file descriptor.....dup2(2)
mmap: map an open file into the processs address space.....mmap(2SEQ)
open: open for reading or writing.....open(2)
open: open for reading or writing.....open(2)
rewinddir, closedir:/ directory: opendir, readdir, telldir, seekdir,.....directory(3X)
a standalone program or bring up the operating system.....boot(8)
run commands performed to stop the operating system.....rc0(1M)
savecore: save a core dump of the operating system.....savecore(1M)
uname: get name of current operating system.....uname(2)
locate: use keywords to identify an operating system command.....locate(1)
assist: assistance using operating system commands.....assist(1)
starter: information about the operating system for beginning users.....starter(1)
help: operating system Help Facility.....help(1)
glossary: definitions of common operating system terms and symbols.....glossary(1)
join: relational database operator.....join(1)
curses: terminal screen handling and optimization package.....curses(3X)
rmvopt: remove a menu option.....rmvopt(1M)
getopt: get option letter from argument vector.....getopt(3C)
addcmdopt: add a command option to a menu.....addcmdopt(1M)
addformopt: add a form/SELECT window option to a menu.....addformopt(1M)
addmenuopt: add a menu option to a menu.....addmenuopt(1M)
object file ldohseek: seek to the optional file header of a common.....ldohseek(3X)
fcntl: file control options.....fcntl(5)
getopt: parse command options.....getopt(1)
getopts, getoptcvt: parse command options.....getopts(1)
stty: set the options for a terminal.....stty(1)
t_optmgmt: manage options for a transport endpoint.....t_optmgmt(3N)
make a directory, or a special or ordinary file.....mknod(2)
architecture information ott: files that hold object.....ott(4)
common assembler and link editor output.....a.out(4)
echo: put string on virtual output.....echo(1F)
fread, fwrite: binary input/ output.....fread(3S)
fprintf, sprintf: print formatted output.....printf(3S)
poll: STREAMS input/ output multiplexing.....poll(2)
/vfprintf, vsprintf: print formatted output of a varargs argument list.....vprintf(3S)
stdio: standard buffered input/ output package.....stdio(3S)

setpgrp: run command in its own process group.....setpgrp(1)
chown: change owner and group of a file.....chown(2)
chown, chgrp: change owner or group.....chown(1)
expand files pack, pcat, unpack: compress and.....pack(1)
makewhatis: generate man page databases.....makealiases(1M)
getpagesize: get system page size.....getpagesize(2SEQ)
4014: paginator for the Tektronix 4014 terminal.....4014(1)
m_fork: execute a subprogram in parallel.....m_fork(3PPS)
lp: Systech parallel line printer interface.....lp(7)
wp: SSM parallel line printer interface.....wp(7)
lpmod: parallel printer canonical processing module.....lpmod(7)
intro: introduction to Parallel Programming Library.....intro(3PPS)
and change virtual memory tuning parameters.....vm_ctl(2SEQ)
tune system-wide virtual-memory parameters.....vmtune(1M)
pmap: display or alter parameters of virtual-to-physical mapping/.....pmap(1M)
get process, process group, and parent process IDs.....getpid(2)
getopt: parse command options.....getopt(1)
getopts, getoptcvt: parse command options.....getopts(1)
cmpt: compatibility partitioning driver.....cmpt(7)
vtoc: VTOC partitioning driver.....vtoc(7)
frexp, ldexp, modf: manipulate parts of floating-point numbers.....frexp(3C)
passmgmt: password files management.....passmgmt(1M)
with information from /etc/ passwd.....pwconv(1M)
password attributes passwd: change login password and.....passwd(1)
passwd: password file.....passwd(4)
getpass: read a password.....getpass(3C)
crypt: password and file encryption functions.....crypt(3X)
passwd: change login password and password attributes.....passwd(1)
passwd: change login password and password attributes.....passwd(1)
pwdhd: daemon to oversee password database creation.....pwdhd(1M)
pwfix: remove incorrectly formatted password entries.....pwfix(1M)
passwd: password file.....passwd(4)
shadow: restricted password file.....shadow(4)
setpwent, endpwent, fgetpwent: get password file entry.....getpwent(3C)
lckpwdf, ulckpwdf: get shadow password file entry.....getspent(3X)
putpwent: write password file entry.....putpwent(3C)
putspent: write shadow password file entry.....putspent(3X)
converts from a two to a one password file scheme.....pwunconv(1M)
passmgmt: password files management.....passmgmt(1M)
mkpwdbm: make a database for fast password lookup.....mkpwdbm(1M)
pwck, grpck: password/group file checkers.....pwck(1M)
files or subsequent lines of one/ paste: merge same lines of several.....paste(1)
bp: binary patcher.....bp(1M)
pathalias: alias file for FACE.....alias(4)
configurable pathname variables pathconf, fpathconf: get.....pathconf(2)
criteria for filename pathconv: search Interpreter.....pathconv(1F)
getcwd: get pathname of current working directory.....getcwd(3C)
fpathconf: get configurable pathname variables.....pathconf(2)
dirname: deliver portions of pathnames.....basename(1)
ncheck: generate pathnames from i-numbers.....ncheck(1M)
sweepon, sweepoff: enable/disable patrol seeks on DCC disks.....sweepon(1M)
grep: search a file for a pattern.....grep(1)

awk: pattern scanning and processing language ..... awk(1)
nawk: pattern scanning and processing language ..... nawk(1)
egrep: search a file for a pattern using full regular/ ..... egrep(1)
regex: match patterns against a string ..... regex(1F)
pause: suspend process until signal ..... pause(2)
pack, pcat, unpack: compress and expand files ..... pack(1)
popen, pclose: initiate pipe to/from a process ..... popen(3S)
pdbx, dbx: parallel/ ..... see *C and Lang Tools Binder*
machid: i386, ns32000, pdp11, u3b, u3b2, u3b5, vax: get processor/ ..... machid(1)
sigpending: examine pending signals ..... sigpending(2)
check the uucp directories and permissions file ..... uucheck(1M)
acct: per-process accounting file format ..... acct(4)
acctcms: command summary from per-process accounting records ..... acctcms(1M)
sys_nerr: system error messages perror, errno, sys_errlist, ... perror(3C)
run a command with priority aging, PFF, swapping, or processor/ ..... noage(1M)
pg: file perusal filter for CRTs ..... pg(1)
virtual address space pmap: map physical addresses into processs ..... pmap(7)
or alter parameters of virtual-to- physical mapping devices ..... pmap(1M)
pipe: create an interprocess channel ..... pipe(2)
sp: STREAMS pipe device ..... sp(7)
tee: pipe fitting ..... tee(1)
popen, pclose: initiate pipe to/from a process ..... popen(3S)
in memory plock: lock process, text, or data ..... plock(2)
plot: graphics interface ..... plot(4)
plot: graphics interface subroutines ..... plot(3X)
virtual-to-physical mapping devices pmap: display or alter parameters of ..... pmap(1M)
processs virtual address space pmap: map physical addresses into ..... pmap(7)
/fpsetsticky: IEEE floating- point environment control ..... fpgetround(3C)
isnand, isnanf: test for floating- point NaN (Not-A-Number) ..... isnan(3C)
ecvt, fcvt, gcvt: convert floating- point number to string ..... ecvt(3C)
modf: manipulate parts of floating- point numbers ..... frexp(3C)
lseek: move read/write file pointer ..... lseek(2)
rewind, ftell: reposition a file pointer in a stream ..... fseek(3S)
multiplexing poll: STREAMS input/output ..... poll(2)
a process popen, pclose: initiate pipe to/from ..... popen(3S)
banner: make posters ..... banner(1)
power, square root/ exp, log, log10, pow, sqrt: exponential, logarithm, ... exp(3M)
power, square root functions ..... exp(3M)
pr: print files ..... pr(1)
/dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, shutacct,/ ..... acctsh(1M)
/lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, shutacct, startup,/ ..... acctsh(1M)
bc: arbitrary- precision arithmetic language ..... bc(1)
atof: convert string to double- precision number ..... strtod(3C)
.environ, .pref, .variables: FACE files ..... env(4)
monitor: prepare execution profile ..... monitor(3C)
cpp: the C language preprocessor ..... see *C and Lang Tools Binder*
factor: obtain the prime factors of a number ..... factor(1)
gps: graphical primitive string, format of graphical files ..... gps(4)
types: primitive system data types ..... types(5)
prs: print an SCCS file ..... prs(1)
date: print and set the date ..... date(1)
cal: print calendar ..... cal(1)

sum: print checksum and block count of a file.....sum(1)
sact: print current SCCS file editing activity .....sact(1)
cat: concatenate and print files.....cat(1)
pr: print files.....pr(1)
printf, fprintf, sprintf: print formatted output.....printf(3S)
vprintf, vfprintf, vsprintf: print formatted output of a varargs/.....vprintf(3S)
of the LP print service lpstat: print information about the status.....lpstat(1)
information about RCS files rlog: print log messages and other.....rlog(1)
object file nm: print name list of.....see *C and Lang Tools Binder*
uname: print name of current system.....uname(1)
news: print news items .....news(1)
infocmp: compare or print out terminfo descriptions.....infocmp(1M)
acctcom: search and print process accounting files .....acctcom(1)
of object files size: print section sizes .....see *C and Lang Tools Binder*
send/cancel requests to the LP print service.....lp(1)
lpadmin: configure the LP print service.....lpadmin(1M)
administer filters used with the LP print service.....lpfilter(1M)
administer forms used with the LP print service.....lpforms(1M)
about the status of the LP print service .....lpstat(1)
/lpshut, lpmove: start/stop the LP print service and move requests.....lpsched(1M)
strace: print STREAMS trace messages .....strace(1M)
prtvtoc: print the VTOC of a block device.....prtvtoc(1M)
prtvtoc: print the VTOC of a standalone disk device .....prtvtoc(8)
id: print user and group IDs and names.....id(1M)
lpmod: parallel printer canonical processing module.....lpmod(7)
lp: Systech parallel line printer interface.....lp(7)
wp: SSM parallel line printer interface .....wp(7)
lptypes: table of printer types supported by the menu system.....lptypes(4)
enable, disable: enable/disable LP printers.....enable(1)
formatted output printf, fprintf, sprintf: print.....printf(3S)
lpusers: set printing queue priorities.....lpusers(1M)
lpusers: set printing queue priorities.....lpusers(1M)
nice: run a command at low priority.....nice(1)
/noswap, onproc: run a command with priority aging, PFF, swapping, or/.....noage(1M)
nice: change priority of a process.....nice(2)
brk, sbrk: change private data segment size.....brk(3PPS)
process attributes proc_ctl: control over various.....proc_ctl(2SEQ)
codestroy: communicate to a process.....coproc(1F)
exit, _exit: terminate process.....exit(2)
fork: create a new process .....fork(2)
inittab: script for the init process.....inittab(4)
kill: terminate a process.....kill(1)
nice: change priority of a process .....nice(2)
pclose: initiate pipe to/from a process.....popen(3S)
wait: await completion of process.....wait(1)
acct: enable or disable process accounting.....acct(2)
acctprc1, acctprc2: process accounting.....acctprc(1M)
acct: per- process accounting file format.....acct(4)
acctcom: search and print process accounting files.....acctcom(1)
acctcms: command summary from per- process accounting records.....acctcms(1M)
alarm: set a process alarm clock.....alarm(2)
times: get process and child process times.....times(2)

proc_ctl: control over various process attributes.....proc_ctl(2SEQ)
m_single, m_multi: mark single- process code section.....m_single(3PPS)
ipcs: report inter- process communication facilities status.....ipcs(1)
init, telinit: process control initialization.....init(1M)
timex: time a command; report process data and system activity.....timex(1)
suspend and resume child process execution.....m_park_procs(3PPS)
setpgrp: run command in its own process group.....setpgrp(1)
/getpgrp, getppid: get process, process group, and parent process IDs.....getpid(2)
setpgrp: set process group ID.....setpgrp(2)
setsid: create session and set process group ID.....setsid(2)
tcgetpgrp: get foreground process group ID.....tcgetpgrp(2)
tcsetpgrp: set foreground process group ID.....tcsetpgrp(2)
m_get_myid: return process identification.....m_get_myid(3PPS)
process, process group, and parent process IDs.....getpid(2)
vfork: spawn new process in a virtual memory efficient way.....vfork(2)
kill: send a signal to a process or a group of processes.....kill(2)
getpid, getpgrp, getppid: get process, process group, and parent/.....getpid(2)
ps: report process status.....ps(1)
plock: lock process, text, or data in memory.....plock(2)
times: get process and child process times.....times(2)
tmp_affinity: bind a process to a processor.....tmp_affinity(2SEQ)
wait, waitpid: wait for a child process to stop or terminate.....wait(2)
ptrace, mptrace: process trace facility.....ptrace(2)
pause: suspend process until signal.....pause(2)
vsig: synchronize a co- process with the controlling FMLI object.....vsig(1F)
checklist: list of filesystems processed by fsck and ncheck.....checklist(4)
a signal to a process or a group of processes.....kill(2)
killall: kill all active processes.....killall(1M)
return number of child processes.....m_get_numprocs(3PPS)
m_kill_procs: kill child processes.....m_kill_procs(3PPS)
m_set_procs: set number of child processes.....m_set_procs(3PPS)
fuser: identify processes using a file or file structure.....fuser(1M)
m4: macro processor.....m4(1)
menu: menu processor.....menu(1M)
tmp_affinity: bind a process to a processor.....tmp_affinity(2SEQ)
tmp_ctl: multi- processor control.....tmp_ctl(2SEQ)
priority aging, PFF, swapping, or processor migration disabled.....noage(1M)
offline_all: take all but one active processor off line.....offline_all(3SEQ)
pdp11, u3b, u3b2, u3b5, vax: get processor type truth value.....machid(1)
online, offline: bring processors online and offline.....online(1M)
mmap: map an open file into the processs address space.....mmap(2SEQ)
munmap: unmap part of the processs address space.....munmap(2SEQ)
pmap: map physical addresses into processs virtual address space.....pmap(7)
product: installed products file.....product(4)
prof: display profile data.....prof(1)
prof: profile within a function.....prof(5)
profil: execution time profile.....profil(2)
monitor: prepare execution profile.....monitor(3C)
profil: execution time profile.....profil(2)
prof: display profile data.....prof(1)
at login time profile: setting up an environment.....profile(4)
prof: profile within a function.....prof(5)

end, etext, edata: last locations in program ..... end(3C)
mt: magnetic tape manipulating program ..... mt(1)
prompt: prompt program ..... prompt(1M)
sdiff: side-by-side difference program ..... sdiff(1)
STREAMS error logger cleanup program ..... strclean(1M)
units: conversion program ..... units(1)
for the uucp file transport program ..... uusched(1M)
assert: verify program assertion ..... assert(3X)
ASSIST menus or command/ astgen: program for generating/modifying ..... astgen(1)
uucico: file transport program for the uucp system ..... uucico(1M)
installboot: installs a boot program on a disk ..... installboot(1M)
boot: run a standalone program or bring up the operating system ..... boot(8)
a standard/restricted command/ programming language ..... ksh(1)
the standard/restricted command programming language ..... sh(1)
intro: introduction to Parallel Programming Library ..... intro(3PPS)
maintenance commands and application programs ..... intro(1)
update, and regenerate groups of programs ..... make(1)
lex: generate programs for simple lexical tasks ..... lex(1)
setlocale: set programs location ..... setlocale(3C)
prompt: prompt program ..... prompt(1M)
prompt: prompt program ..... prompt(1M)
proto: prototype job file for at ..... proto(4)
t_getinfo: get protocol-specific service information ..... t_getinfo(3N)
proto: prototype job file for at ..... proto(4)
prs: print an SCCS file ..... prs(1)
/monacct, nulladm, prctmp, prdaily, prtacct, shutacct, startup/ ..... acctsh(1M)
device prtvtoc: print the VTOC of a block ..... prtvtoc(1M)
standalone disk device prtvtoc: print the VTOC of a ..... prtvtoc(8)
ps: report process status ..... ps(1)
streampty: STREAMS pseudo terminal driver ..... streampty(7)
getpseudotty: get pseudo terminals ..... getpseudotty(3SEQ)
cn: pseudoconsole driver ..... cn(7)
generate uniformly distributed pseudorandom numbers ..... drand48(3C)
time, ptime: time a command ..... time(1)
facility ptrace, mptrace: process trace ..... ptrace(2)
uuto, uupick: public UNIX-to-UNIX system file copy ..... uuto(1C)
ungetc: push character back into input stream ..... ungetc(3S)
puts, fputs: put a string on a stream ..... puts(3S)
putc, putchar, fputc, putw: put character or word on a stream ..... putc(3S)
getdents: read directory entries and put in a file system independent/ ..... getdents(2)
echo: put string on virtual output ..... echo(1F)
character or word on a stream putc, putchar, fputc, putw: put ..... putc(3S)
or word on a stream putc, putchar, fputc, putw: put character ..... putc(3S)
environment putenv: change or add value to ..... putenv(3C)
putmsg: send a message on a stream ..... putmsg(2)
putpwent: write password file entry ..... putpwent(3C)
stream puts, fputs: put a string on a ..... puts(3S)
message: puts its arguments on message line ..... message(1F)
entry putspent: write shadow password file ..... putspent(3X)
getut: getutent, getutid, getutline, pututline, setutent, endutent/ ..... getut(3C)
putc, putchar, fputc, putw: put character or word on a stream ..... putc(3S)
checkers pwck, grpck: password/group file ..... pwck(1M)

/etc/shadow with information from/ pwconv: installs and updates ..... pwconv(1M)
pwd: working directory name ..... pwd(1)
database creation pwdhd: daemon to oversee password ..... pwdhd(1M)
password entries pwfix: remove incorrectly formatted ..... pwfix(1M)
one password file scheme pwunconv: converts from a two to a ..... pwunconv(1M)
qsort: quicker sort ..... qsort(3C)
tput: initialize a terminal or query terminfo database ..... tput(1)
route user message to menu message queue ..... menumsg(1M)
msgget: get message queue ..... msgget(2)
lpusers: set printing queue priorities ..... lpusers(1M)
ipcrm: remove a message queue, semaphore set, or shared memory ID ..... ipcrm(1)
quotacheck: filesystem quota consistency checker ..... quotacheck(1M)
quota: display disk usage and limits ..... quota(1)
consistency checker quotacheck: filesystem quota ..... quotacheck(1M)
quotactl: manipulate disk quotas ..... quotactl(2SEQ)
quotaon, quotaoff: turn filesystem quotas on and off ..... quotaon(1M)
quotas on and off quotaon, quotaoff: turn filesystem ..... quotaon(1M)
edquota: edit user quotas ..... edquota(1M)
quotactl: manipulate disk quotas ..... quotactl(2SEQ)
repquota: summarize quotas for a filesystem ..... repquota(1M)
quotaon, quotaoff: turn filesystem quotas on and off ..... quotaon(1M)
generator rand, srand: simple random-number ..... rand(3C)
fsirand: install random inode generation numbers ..... fsirand(1M)
rand, srand: simple random-number generator ..... rand(3C)
fsplit: split f77, ratfor, or efl files ..... fsplit(1)
ratfor: rational Fortran dialect ..... ratfor(1)
ratfor: rational Fortran dialect ..... ratfor(1)
scripts rc: reboot and shutdown command ..... rc(4)
the operating system rc0: run commands performed to stop ..... rc0(1M)
multi-user environment rc2: run commands performed for ..... rc2(1M)
rcs: change RCS file attributes ..... rcs(1)
rcsintro: introduction to RCS commands ..... rcsintro(1)
rcsfile: format of RCS file ..... rcsfile(4)
rcs: change RCS file attributes ..... rcs(1)
sccstorcs: build RCS file from SCCS file ..... sccstorcs(1)
messages and other information about RCS files ..... rlog(1)
ci: check in RCS revisions ..... ci(1)
co: check out RCS revisions ..... co(1)
rcsdiff: compare RCS revisions ..... rcsdiff(1)
rcsmerge: merge RCS revisions ..... rcsmerge(1)
rcsdiff: compare RCS revisions ..... rcsdiff(1)
rcsfile: format of RCS file ..... rcsfile(4)
commands rcsintro: introduction to RCS ..... rcsintro(1)
rcsmerge: merge RCS revisions ..... rcsmerge(1)
getpass: read a password ..... getpass(3C)
of a common object file ldtbread: read an indexed symbol table entry ..... ldtbread(3X)
header of a/ ldshread, ldnshread: read an indexednamed section ..... ldshread(3X)
structure read_constab: read constab entries into a ..... read_constab(3SEQ)
file system independent/ getdents: read directory entries and put in a ..... getdents(2)
read: read from file ..... read(2)
mail, rmail: send mail to users or read mail ..... mail(1)
line: read one line ..... line(1)

read: read from file ..... read(2)
of an archive file ldahread: read the archive header of a member ..... ldahread(3X)
object file ldfhread: read the file header of a common ..... ldfhread(3X)
readlink: read value of a symbolic link ..... readlink(2SEQ)
readv: read vector from file ..... readv(2SEQ)
into a structure read_constab: read constab entries ..... read_constab(3SEQ)
directory: opendir, readdir, telldir, seekdir, rewinddir/ ..... directory(3X)
gets longest line readfile, longline: reads file and ..... readfile(1F)
open a common object file for reading ..... ldopen(3X)
open: open for reading or writing ..... open(2)
link readlink: read value of a symbolic ..... readlink(2SEQ)
readfile, longline: reads file and gets longest line ..... readfile(1F)
readv: read vector from file ..... readv(2SEQ)
lseek: move read/write file pointer ..... lseek(2)
tirdwr: Transport Interface read/write interface STREAMS module ..... tirdwr(7)
/get real user, effective user, real group, and effective group IDs ..... getuid(2)
/geteuid, getgid, getegid: get real user, effective user, real/ ..... getuid(2)
malloc, free, realloc, calloc: main memory allocator ..... malloc(3C)
fast main memory/ malloc, free, realloc, calloc, mallopt, mallinfo: ..... malloc(3X)
rc: reboot and shutdown command scripts ..... rc(4)
t_rcvudata: receive a data unit ..... t_rcvudata(3N)
t_rcvuderr: receive a unit data error indication ..... t_rcvuderr(3N)
over a connection t_rcv: receive data or expedited data sent ..... t_rcv(3N)
connect request t_rcvconnect: receive the confirmation from a ..... t_rcvconnect(3N)
lockf: record locking on files ..... lockf(3C)
ed, red: text editor ..... ed(1)
setcolor: redefine or create a color ..... setcolor(1F)
dltsectbl: refresh security table ..... dltsectbl(1M)
regular expression regcmp, regex: compile and execute ..... regcmp(3X)
regcmp: regular expression compile ..... regcmp(1)
make: maintain, update, and regenerate groups of programs ..... make(1)
expression regcmp, regex: compile and execute regular ..... regcmp(3X)
string regex: match patterns against a ..... regex(1F)
and match routines regexp: regular expression compile ..... regexp(5)
regcmp, regex: compile and execute regular expression ..... regcmp(3X)
regcmp: regular expression compile ..... regcmp(1)
routines regexp: regular expression compile and match ..... regexp(5)
a file for a pattern using full regular expressions ..... egrep(1)
reinit: run an initialization file ..... reinit(1F)
accept, reject: allow or prevent LP requests ..... accept(1M)
comm: select or reject lines common to two sorted files ..... comm(1)
join: relational database operator ..... join(1)
t_sndrel: initiate an orderly release ..... t_sndrel(3N)
acknowledge receipt of an orderly release indication ..... t_rcvrel(3N)
common object file reloc: relocation information for a ..... reloc(4)
ldrseek, ldnrseek: seek to relocation entries of a section of a common/ ..... ldrseek(3X)
reloc: relocation information for a common object file ..... reloc(4)
/ceil, fmod, fabs: floor, ceiling, remainder, absolute value functions ..... floor(3M)
calendar: reminder service ..... calendar(1)
uuxqt: execute remote command requests ..... uuxqt(1M)
Uutry: try to contact remote system with debugging on ..... uutry(1M)
ct: spawn getty to a remote terminal ..... ct(1C)

rmdel: remove a delta from an SCCS file.....rmdel(1)
rmdir: remove a directory.....rmdir(2)
rmvopt: remove a menu option.....rmvopt(1M)
set, or shared memory ID ipcrm: remove a message queue, semaphore.....ipcrm(1)
unlink: remove directory entry.....unlink(2)
rm, rmdir: remove files or directories.....rm(1)
password entries pwfix: remove incorrectly formatted.....pwfix(1M)
deroff: remove nroff, tbl, and eqn constructs.....deroff(1)
rename: change the name of a file.....rename(2)
consistency check and interactive repair.....fsck(1M)
uniq: report repeated lines in a file.....uniq(1)
clock: report CPU time used.....clock(3C)
fsstat: report filesystem status.....fsstat(1M)
facilities status ipcs: report inter-process communication.....ipcs(1)
df: report number of free disk blocks and inodes.....df(1M)
mailbug: submit a bug report or enhancement request.....mailbug(1)
sar: sa1, sa2, sadc: system activity report package.....sar(1M)
timex: time a command; report process data and system activity.....timex(1)
ps: report process status.....ps(1)
uniq: report repeated lines in a file.....uniq(1)
fseek, rewind, ftell: reposition a file pointer in a stream.....fseek(3S)
filesystem repquota: summarize quotas for a.....repquota(1M)
submit a bug report or enhancement request.....mailbug(1)
t_accept: accept a connect request.....t_accept(3N)
t_listen: listen for a connect request.....t_listen(3N)
the confirmation from a connect request.....t_rcvconnect(3N)
send user-initiated disconnect request.....t_snddis(3N)
accept, reject: allow or prevent LP requests.....accept(1M)
the LP print service and move requests.....lpsched(1M)
uuxqt: execute remote command requests.....uuxqt(1M)
lp, cancel: send/cancel requests to the LP print service.....lp(1)
to its default values reset: reset the current form field.....reset(1F)
default values reset: reset the current form field to its.....reset(1F)
get_process_stats: get resource utilization info.....get_process_stats(2SEQ)
restore: incremental filesystem restore.....restore(1M)
restore restore: incremental filesystem.....restore(1M)
/ksh, rksh – KornShell, a standard/ restricted command and programming/.....ksh(1)
sh, rsh: shell, the standard/ restricted command programming/.....sh(1)
shadow: restricted password file.....shadow(4)
/m_rele_procs: suspend and resume child process execution.....m_park_procs(3PPS)
usage examples usage: retrieve a command description and.....usage(1)
t_rcvdis: retrieve information from disconnect.....t_rcvdis(3N)
object file symbol table/ ldgetname: retrieve symbol name for common.....ldgetname(3X)
abs: return integer absolute value.....abs(3C)
logname: return login name of user.....logname(3X)
m_get_numprocs: return number of child procs.....m_get_numprocs(3PPS)
m_get_myid: return process identification.....m_get_myid(3PPS)
getenv: return value for environment name.....getenv(3C)
stat: data returned by stat system call.....stat(5)
marked menu items getitems: returns a list of the currently.....getitems(1F)
getfrm: returns the current frame number.....getfrm(1F)
cpus_online: returns the number of CPUs on-line.....cpus_online(3PPS)

col: filter reverse line-feeds.....col(1)
ci: check in RCS revisions.....ci(1)
co: check out RCS revisions.....co(1)
rcsdiff: compare RCS revisions.....rcsdiff(1)
rcsmerge: merge RCS revisions.....rcsmerge(1)
pointer in a stream fseek, rewind, ftell: reposition a file.....fseek(3S)
/opendir, readdir, telldir, seekdir, rewinddir, closedir: directory/.....directory(3X)
creat: create a new file or rewrite an existing one.....creat(2)
restricted command/ ksh, rksh: KornShell, a standard/.....ksh(1)
information about RCS files rlog: print log messages and other.....rlog(1)
directories rm, rmdir: remove files or.....rm(1)
mail, rmail: send mail to users or read mail.....mail(1)
file rmdel: remove a delta from an SCCS.....rmdel(1)
rmdir: remove a directory.....rmdir(2)
rm, rmdir: remove files or directories.....rm(1)
rmvopt: remove a menu option.....rmvopt(1M)
chroot: change root directory.....chroot(2)
chroot: change root directory for a command.....chroot(1M)
logarithm, power, square root functions.....exp(3M)
queue menumsg: route user message to menu message.....menumsg(1M)
ldfcn: common object file access routines.....ldfcn(4)
regular expression compile and match routines.....regexp(5)
programming language sh, rsh: shell, standard/restricted command/.....sh(1)
nice: run a command at low priority.....nice(1)
nohup: run a command immune to hangups and quits.....nohup(1)
shell: run a command using shell.....shell(1F)
noage, nopff, noswap, onproc: run a command with priority aging, PFF/.....noage(1M)
the operating system boot: run a standalone program or bring up.....boot(8)
run: run an executable.....run(1F)
edc: run an externally described command.....edc(1M)
reinit: run an initialization file.....reinit(1F)
setpgrp: run command in its own process group.....setpgrp(1)
multi-user environment rc2: run commands performed for.....rc2(1M)
operating system rc0: run commands performed to stop the.....rc0(1M)
runacct: run daily accounting.....runacct(1M)
run: run an executable.....run(1F)
runacct: run daily accounting.....runacct(1M)
Administration SA: devices administered by System.....sa(7)
sar: sa1, sa2, sadc: system activity report package.....sar(1M)
sar: sa1, sa2, sadc: system activity report package.....sar(1M)
editing activity sact: print current SCCS file.....sact(1)
sar: sa1, sa2, sadc: system activity report package.....sar(1M)
report package sar: sa1, sa2, sadc: system activity.....sar(1M)
sar: system activity reporter.....sar(1)
savecore: save a core dump of the operating system.....savecore(1M)
operating system savecore: save a core dump of the.....savecore(1M)
brk, sbrk: change data segment space allocation.....brk(2)
brk, sbrk: change private data segment size.....brk(3PPS)
formatted input scanf, fscanf, sscanf: convert.....scanf(3S)
bfs: big file scanner.....bfs(1)
awk: pattern scanning and processing language.....awk(1)
nawk: pattern scanning and processing language.....nawk(1)

change the delta commentary of an SCCS delta ..... cdc(1)
comb: combine SCCS deltas ..... comb(1)
delta: make a delta (change) to an SCCS file ..... delta(1)
get: get a version of an SCCS file ..... get(1)
prs: print an SCCS file ..... prs(1)
rmdel: remove a delta from an SCCS file ..... rmdel(1)
sccsdiff: compare two versions of an SCCS file ..... sccsdiff(1)
sccsfile: format of SCCS file ..... sccsfile(4)
sccstorcs: build RCS file from SCCS file ..... sccstorcs(1)
unget: undo a previous get of an SCCS file ..... unget(1)
val: validate SCCS file ..... val(1)
sact: print current SCCS file editing activity ..... sact(1)
admin: create and administer SCCS files ..... admin(1)
what: identify SCCS files ..... what(1)
SCCS file sccsdiff: compare two versions of an ..... sccsdiff(1)
sccsfile: format of SCCS file ..... sccsfile(4)
file sccstorcs: build RCS file from SCCS ..... sccstorcs(1)
board SCED: SCSI/Ethernet/Diagnostics ..... sced(7)
ckbupscd: check filesystem backup schedule ..... ckbupscd(1M)
transport program uusched: the scheduler for the uucp file ..... uusched(1M)
unlock locks s_init_lock, s_lock, s_clock, s_unlock: initialize, lock, ... s_lock(3PPS)
object file scnhdr: section header for a common ..... scnhdr(4)
image file scr_dump: format of curses screen ..... scr_dump(4)
more: view a file one full screen at a time ..... more(1)
curses: terminal screen handling and optimization package ..... curses(3X)
scr_dump: format of curses screen image file ..... scr_dump(4)
vi: screen-oriented (visual) display editor based on ex ..... vi(1)
inittab: script for the init process ..... inittab(4)
rc: reboot and shutdown command scripts ..... rc(4)
tg: SCSI 1/2-inch magnetic tape interface ..... tg(7)
sd: SCSI disk controller ..... sd(7)
wd: SSM SCSI disk controller ..... wd(7)
CCSformat: standalone embedded SCSI disk formatter ..... ccsformat(8)
tm: SCSI tape controller ..... tm(7)
ts: SCSI tape controller ..... ts(7)
SCED: SCSI/Ethernet/Diagnostics board ..... sced(7)
getscsiinfo, getscsimatch: get scsiinfo structure ..... getscsiinfo(3SEQ)
sd: SCSI disk controller ..... sd(7)
program sdiff: side-by-side difference ..... sdiff(1)
fgrep: search a file for a character string ..... fgrep(1)
grep: search a file for a pattern ..... grep(1)
full regular expressions egrep: search a file for a pattern using ..... egrep(1)
bsearch: binary search a sorted table ..... bsearch(3C)
acctcom: search and print process accounting files ..... acctcom(1)
lsearch, lfind: linear search and update ..... lsearch(3C)
pathconv: search Interpreter criteria for filename ..... pathconv(1F)
hcreate, hdestroy: manage hash search tables ..... hsearch(3C)
tfind, tdelete, twalk: manage binary search trees ..... tsearch(3C)
manipulate the object file comment section ..... mcs(1)
m_multi: mark single-process code section ..... m_single(3PPS)
scnhdr: section header for a common object file ..... scnhdr(4)
/ldnshread: read an indexednamed section header of a common object/ ..... ldshread(3X)

/seek to line number entries of a section of a common object file ..... ldlseek(3X)
/seek to relocation entries of a section of a common object file ..... ldrseek(3X)
/ldnsseek: seek to an indexednamed section of a common object file ..... ldsseek(3X)
dltsectbl: refresh security table ..... dltsectbl(1M)
task: global security table ..... task(4)
tasktbl: create security table in shared memory ..... tasktbl(1M)
sed: stream editor ..... sed(1)
/nrand48, mrand48, jrand48, srand48, seed48, lcong48: generate uniformly/ ..... drand48(3C)
a common object/ ldsseek, ldnsseek: seek to an indexednamed section of ..... ldsseek(3X)
section of a/ ldlseek, ldnlseek: seek to line number entries of a ..... ldlseek(3X)
ldrseek, ldnrseek: seek to relocation entries of a section of a/ ..... ldrseek(3X)
a common object file ldohseek: seek to the optional file header of ..... ldohseek(3X)
object file ldtbseek: seek to the symbol table of a common ..... ldtbseek(3X)
/opendir, readdir, telldir, seekdir, rewinddir, closedir/ ..... directory(3X)
sweepoff: enable/disable patrol seeks on DCC disks ..... sweepon(1M)
shmget: get shared memory segment identifier ..... shmget(2)
brk, sbrk: change private data segment size ..... brk(3PPS)
shbrk, shsbrk: change shared data segment size ..... shbrk(3PPS)
brk, sbrk: change data segment space allocation ..... brk(2)
sorted files comm: select or reject lines common to two ..... comm(1)
greek: select terminal filter ..... greek(1)
addformopt: add a form/ SELECT window option to a menu ..... addformopt(1M)
semctl: semaphore control operations ..... semctl(2)
semop: semaphore operations ..... semop(2)
ipcrm: remove a message queue, semaphore set, or shared memory ID ..... ipcrm(1)
semget: get set of semaphores ..... semget(2)
semctl: semaphore control operations ..... semctl(2)
semget: get set of semaphores ..... semget(2)
semop: semaphore operations ..... semop(2)
t_sndudata: send a data unit ..... t_sndudata(3N)
putmsg: send a message on a stream ..... putmsg(2)
kill: send a signal to a process or a group of processes ..... kill(2)
tfnotify: send completion message ..... tfnotify(1M)
connection t_snd: send data or expedited data over a ..... t_snd(3N)
mail, rmail: send mail to users or read mail ..... mail(1)
t_snddis: send user-initiated disconnect request ..... t_snddis(3N)
lp, cancel: send/cancel requests to the LP print service ..... lp(1)
receive data or expedited data sent over a connection ..... t_rcv(3N)
intro: introduction to Sequent system calls ..... intro(2SEQ)
dis386: disassemble Series 386 instructions ..... dis386(3SEQ)
SSM: System Services Module ..... ssm(7)
setsid: create session and set process group ID ..... setsid(2)
ascii: map of ASCII character set ..... ascii(5)
alarm: set a process alarm clock ..... alarm(2)
umask: set and get file creation mask ..... umask(2)
stty, gtty: set and get terminal state (defunct) ..... stty(2)
in core or in files set, unset: set and unset environment variables ..... set(1F)
sigstack: set and/or get signal stack context ..... sigstack(2SEQ)
timezone: set default system time zone ..... timezone(4)
backup.info: backup set definitions ..... backup_info(4)
setdtablesize: set descriptor table size ..... setdtablesize(2SEQ)
env: set environment for command execution ..... env(1)

utime: set file access and modification times.....utime(2)
umask: set file-creation mode mask.....umask(1)
tcsetpgrp: set foreground process group ID.....tcsetpgrp(2)
setgroups: set group access list.....setgroups(2SEQ)
cpset: copy file and set mode, user, and group IDs.....cpset(1)
m_set_procs: set number of child processes.....m_set_procs(3PPS)
semget: get set of semaphores.....semget(2)
remove a message queue, semaphore set, or shared memory ID.....ipcrm(1)
lpusers: set printing queue priorities.....lpusers(1M)
setpgrp: set process group ID.....setpgrp(2)
setsid: create session and set process group ID.....setsid(2)
setlocale: set programs location.....setlocale(3C)
tabs: set tabs on a terminal.....tabs(1)
tcgetattr, tcsetattr: get/ set terminal state.....tcgetattr(2)
line discipline getty: set terminal type, modes, speed, and.....getty(1M)
line discipline uugetty: set terminal type, modes, speed, and.....uugetty(1M)
date: print and set the date.....date(1)
stty: set the options for a terminal.....stty(1)
stime: set time.....stime(2)
environment variables in core or in/ set, unset: set and unset.....set(1F)
setuid, setgid: set user and group IDs.....setuid(2)
ulimit: get and set user limits.....ulimit(2)
a stream setbuf, setvbuf: assign buffering to.....setbuf(3S)
/toascci, _tolower, _toupper, setchrclass: character handling.....ctype(3C)
setcolor: redefine or create a color.....setcolor(1F)
size setdtablesize: set descriptor table.....setdtablesize(2SEQ)
setuid, setgid: set user and group IDs.....setuid(2)
getgrent, getgrgid, getgrnam, setgrent, endgrent, fgetgrent: get group/.....getgrent(3C)
setgroups: set group access list.....setgroups(2SEQ)
setjmp, longjmp: non-local goto.....setjmp(3C)
crypt, setkey, encrypt: generate hashing encryption.....crypt(3C)
setlocale: set programs location.....setlocale(3C)
setmnt: establish mount table.....setmnt(1M)
/endmntent, getmntent, hasmntopt, setmntent: get file system/.....getmntent(3SEQ)
setpgid: manipulate signal sets.....setpgid(2)
process group setpgrp: run command in its own.....setpgrp(1)
setpgrp: set process group ID.....setpgrp(2)
getpwent, getpwuid, getpwnam, setpwent, endpwent, fgetpwent: get/.....getpwent(3C)
process group ID setsid: create session and set.....setsid(2)
getspent, getspnam, setspent, endspent, fgetspent, lckpwdf/.....getspent(3X)
profile: setting up an environment at login time.....profile(4)
gettydefs: speed and terminal settings used by getty.....gettydefs(4)
IDs setuid, setgid: set user and group.....setuid(2)
user setup: initialize system for first.....setup(1)
/getutid, getutline, pututline, setutent, endutent, utmpname/.....getut(3C)
setbuf, setvbuf: assign buffering to a stream.....setbuf(3S)
machine-independent fashion sputl, sgetl: access long integer data in a.....sputl(3X)
standard/restricted command/ sh, rsh: shell, the.....sh(1)
/fgetspent, lckpwdf, ulckpwdf: get shadow password file entry.....getspent(3X)
putspent: write shadow password file entry.....putspent(3X)
shadow: restricted password file.....shadow(4)
pwconv: installs and updates /etc/ shadow with information from/.....pwconv(1M)

shbrk, shsbrk: change — shared data segment size ..... shbrk(3PPS)
tasktbl: create security table in — shared memory ..... tasktbl(1M)
shmalloc, shrealloc, shfree: — shared memory allocator ..... shmalloc(3PPS)
shmctl: — shared memory control operations ..... shmctl(2)
a message queue, semaphore set, or — shared memory ID ..... ipcrm(1)
shmop, shmat, shmdt: — shared memory operations ..... shmop(2)
shmget: get — shared memory segment identifier ..... shmget(2)
segment size — shbrk, shsbrk: change shared data ..... shbrk(3PPS)
shell: run a command using — shell ..... shell(1F)
assign menu system as user login — shell ..... tfmenush(1M)
system: issue a — shell command ..... system(3S)
csh: a — shell (command interpreter) with C-like syntax ..... csh(1)
/shutacct, startup, turnacct: — shell procedures for accounting ..... acctsh(1M)
shell: run a command using shell ..... shell(1F)
shells: — shell table ..... shells(4)
command programming/ sh, rsh: — shell, the standard/restricted ..... sh(1)
shells: shell table ..... shells(4)
shmalloc, shrealloc, — shfree: shared memory allocator ..... shmalloc(3PPS)
memory allocator — shmalloc, shrealloc, shfree: shared ..... shmalloc(3PPS)
shmop, — shmat, shmdt: shared memory operations ..... shmop(2)
operations — shmctl: shared memory control ..... shmctl(2)
shmop, shmat, — shmdt: shared memory operations ..... shmop(2)
identifier — shmget: get shared memory segment ..... shmget(2)
operations — shmop, shmat, shmdt: shared memory ..... shmop(2)
the machine — showcfg: show the configuration of ..... showcfg(1M)
allocator shmalloc, — shrealloc, shfree: shared memory ..... shmalloc(3PPS)
shbrk, — shsbrk: change shared data segment size ..... shbrk(3PPS)
shutdown: — shut down system, change system state ..... shutdown(1M)
/nulladm, prctmp, prdaily, prtacct, — shutacct, startup, turnacct: shell/ ..... acctsh(1M)
rc: reboot and — shutdown command scripts ..... rc(4)
shutdown: system — shutdown information file ..... shutdown(4)
system state — shutdown: shut down system, change ..... shutdown(1M)
information file — shutdown: system shutdown ..... shutdown(4)
actions — sigaction: examine and change signal ..... sigaction(2)
manipulate/ sigemptyset, sigfillset, — sigaddset, sigdelset, sigismember: ..... sigemptyset(2)
sigcontext: get signal context ..... sigcontext(2SEQ)
sigemptyset, sigfillset, sigaddset, — sigdelset, sigismember: manipulate/ ..... sigemptyset(2)
sigdelset, sigismember: manipulate/ — sigemptyset, sigfillset, sigaddset, ... sigemptyset(2)
sigismember:/ sigemptyset, — sigfillset, sigaddset, sigdelset, ... sigemptyset(2)
sigpause: signal management sigset, — sighold, sigrelse, sigignore, ... sigset(2)
sigset, sighold, sigrelse, — sigignore, sigpause: signal/ ..... sigset(2)
/sigfillset, sigaddset, sigdelset, — sigismember: manipulate signal sets ..... sigemptyset(2)
sigsetjmp, — siglongjmp: non-local jumps ..... sigsetjmp(3C)
login: — sign on ..... login(1)
pause: suspend process until — signal ..... pause(2)
specify what to do upon receipt of a — signal ..... signal(2)
sigsuspend: wait for a — signal ..... sigsuspend(2)
sigaction: examine and change — signal actions ..... sigaction(2)
sigcontext: get — signal context ..... sigcontext(2SEQ)
sigrelse, sigignore, sigpause: — signal management ..... sigset(2)
setpgid: manipulate — signal sets ..... setpgid(2)
sigdelset, sigismember: manipulate — signal sets ..... sigemptyset(2)

receipt of a signal — signal: specify what to do upon.....signal(2)
sigstack: set and/or get — signal stack context.....sigstack(2SEQ)
kill: send a — signal to a process or a group of processes.....kill(2)
sigpending: examine pending — signals.....sigpending(2)
examine and change blocked — signals.....sigprocmask(2)
ssignal, gsignal: software — signals.....ssignal(3C)
sighold, sigrelse, sigignore, — sigpause: signal management.....sigset(2)
— sigpending: examine pending signals.....sigpending(2)
blocked signals — sigprocmask: examine and change.....sigprocmask(2)
signal management sigset, sighold, — sigrelse, sigignore, sigpause:.....sigset(2)
sigignore, sigpause: signal/ — sigset, sighold, sigrelse,.....sigset(2)
jumps — sigsetjmp, siglongjmp: non-local.....sigsetjmp(3C)
stack context — sigstack: set and/or get signal.....sigstack(2SEQ)
— sigsuspend: wait for a signal.....sigsuspend(2)
atan2: trigonometric/ trig: — sin, cos, tan, asin, acos, atan,.....trig(3M)
m_single, m_multi: mark — single-process code section.....m_single(3PPS)
functions — sinh, cosh, tanh: hyperbolic.....sinh(3M)
initialize barrier, wait at barrier — s_init_barrier, s_wait_barrier:.....s_wait_barrier(3PPS)
s_unlock: initialize, lock, unlock/ — s_init_lock, s_lock, s_clock,.....s_lock(3PPS)
sbrk: change private data segment — size.....brk(3PPS)
getdtablesize: get descriptor table — size.....getdtablesize(2SEQ)
getpagesize: get system page — size.....getpagesize(2SEQ)
setdtablesize: set descriptor table — size.....setdtablesize(2SEQ)
shsbrk: change shared data segment — size.....shbrk(3PPS)
of object files — size: print section sizes.....see *C and Lang Tools Binder*
interval — sleep: suspend execution for an.....sleep(1)
interval — sleep: suspend execution for.....sleep(3C)
s_init_lock, — s_lock, s_clock, s_unlock: initialize, lock,/.....s_lock(3PPS)
ttyslot: find the — slot in the utmp file of the current user.....ttyslot(3C)
consmem, — sm, ss: console memory drivers.....consmem(7)
ssignal, gsignal: — software signals.....ssignal(3C)
qsort: quicker — sort.....qsort(3C)
tsort: topological — sort.....tsort(1)
sort: — sort and/or merge files.....sort(1)
— sort: sort and/or merge files.....sort(1)
select or reject lines common to two — sorted files.....comm(1)
bsearch: binary search a — sorted table.....bsearch(3C)
an object file list: produce C — source listing from.....see *C and Lang Tools Binder*
— sp: STREAMS pipe device.....sp(7)
open file into the processs address — space.....mmap(2SEQ)
unmap part of the processs address — space.....munmap(2SEQ)
into processs virtual address — space.....pmap(7)
brk, sbrk: change data segment — space allocation.....brk(2)
ct: — spawn getty to a remote terminal.....ct(1C)
memory efficient way vfork: — spawn new process in a virtual.....vfork(2)
fspec: format — specification in text files.....fspec(4)
signal: — specify what to do upon receipt of a signal.....signal(2)
getty: set terminal type, modes, — speed, and line discipline.....getty(1M)
uugetty: set terminal type, modes, — speed, and line discipline.....uugetty(1M)
gettydefs: — speed and terminal settings used by getty.....gettydefs(4)
find spelling errors — spell, hashmake, spellin, hashcheck:.....spell(1)
file — spellhist: unmatched spelling words.....spellhist(4)

spell, hashmake, spellin, hashcheck: find spelling errors ..... spell(1)
hashmake, spellin, hashcheck: find spelling errors ..... spell(1)
spellhist: unmatched spelling words file ..... spellhist(4)
csplit: context split ..... csplit(1)
split: split a file into pieces ..... split(1)
fsplit: split f77, ratfor, or efl files ..... fsplit(1)
split: split a file into pieces ..... split(1)
uucleanup: uucp spool directory clean-up ..... uucleanup(1M)
printf, fprintf, sprintf: print formatted output ..... printf(3S)
data in a machine-independent/ sputl, sgetl: access long integer ..... sputl(3X)
square root/ exp, log, log10, pow, sqrt: exponential, logarithm, power, ... exp(3M)
sqrt: exponential, logarithm, power, square root functions ..... exp(3M)
rand, srand: simple random-number generator ..... rand(3C)
/lrand48, nrand48, mrand48, jrand48, srand48, seed48, lcong48: generate/ ..... drand48(3C)
consmem, sm, ss: console memory drivers ..... consmem(7)
scanf, fscanf, sscanf: convert formatted input ..... scanf(3S)
ssignal, gsignal: software signals ..... ssignal(3C)
wp: SSM parallel line printer interface ..... wp(7)
wd: SSM SCSI disk controller ..... wd(7)
SSM: System Services Module ..... ssm(7)
terminal multiplexor driver st: Systech MTI-800/1600/1650 ..... st(7)
sigstack: set and/or get signal stack context ..... sigstack(2SEQ)
copy: perform a standalone data transfer ..... copy(8)
prtvtoc: print the VTOC of a standalone disk device ..... prtvtoc(8)
dual-channel disk/ zdformat: standalone disk formatter for ..... zdformat(8)
CCSformat: standalone embedded SCSI disk formatter ..... ccsformat(8)
dump: standalone memory dump ..... dump(8)
operating system boot: run a standalone program or bring up the ..... boot(8)
intro: introduction to system standalone utilities ..... intro(8)
stdio: standard buffered input/output package ..... stdio(3S)
package stdipc: ftok: standard interprocess communication ..... stdipc(3C)
sh, rsh: shell, the standard/restricted command programming/ ..... sh(1)
operating system for beginning/ starter: information about the ..... starter(1)
lpsched, lpshut, lpmove: start/stop the LP print service and move/ ..... lpsched(1M)
/prctmp, prdaily, prtacct, shutacct, startup, turnacct: shell procedures/ ..... acctsh(1M)
call stat: data returned by stat system ..... stat(5)
stat, lstat, fstat: get file status ..... stat(2)
stat: data returned by stat system call ..... stat(5)
information statfs, fstatfs: get filesystem ..... statfs(2)
ustat: get filesystem statistics ..... ustat(2)
ff: list filenames and statistics for a filesystem ..... ff(1M)
fsstat: report filesystem status ..... fsstat(1M)
communication facilities status ..... ipcs(1)
ps: report process status ..... ps(1)
stat, lstat, fstat: get file status ..... stat(2)
feof, clearerr, fileno: stream status inquiries ..... ferror(3S)
uustat: uucp status inquiry and job control ..... uustat(1C)
lpstat: print information about the status of the LP print service ..... lpstat(1)
input/output package stdio: standard buffered ..... stdio(3S)
communication package stdipc: ftok: standard interprocess ..... stdipc(3C)
stime: set time ..... stime(2)
waitpid: wait for a child process to stop or terminate ..... wait(2)

lpsched, lpshut, lpmove: start/ stop the LP print service and move/.....lpsched(1M)
rc0: run commands performed to stop the operating system.....rc0(1M)
icheck: filesystem storage consistency check.....icheck(1M)
strace: print STREAMS trace messages.....strace(1M)
strncmp, strcpy, strncpy,/ string: strcat, strdup, strncat, strcmp,.....string(3C)
/strncmp, strcpy, strncpy, strlen, strchr, strrchr, strpbrk, strspn,/.....string(3C)
cleanup program strclean: STREAMS error logger.....strclean(1M)
string: strcat, strdup, strncat, strcmp, strncmp, strcpy, strncpy,/.....string(3C)
/strdup, strncat, strcmp, strncmp, strcpy, strncpy, strlen, strchr,/.....string(3C)
/strchr, strrchr, strpbrk, strspn, strcspn, strtok: string operations.....string(3C)
strcpy, strncpy,/ string: strcat, strdup, strncat, strcmp, strncmp,.....string(3C)
fclose, fflush: close or flush a stream.....fclose(3S)
fopen, freopen, fdopen: open a stream.....fopen(3S)
reposition a file pointer in a stream.....fseek(3S)
getw: get character or word from a stream.....getc(3S)
getmsg: get next message off a stream.....getmsg(2)
gets, fgets: get a string from a stream.....gets(3S)
putw: put character or word on a stream.....putc(3S)
putmsg: send a message on a stream.....putmsg(2)
puts, fputs: put a string on a stream.....puts(3S)
setvbuf: assign buffering to a stream.....setbuf(3S)
push character back into input stream.....ungetc(3S)
sed: stream editor.....sed(1)
ferror, feof, clearerr, fileno: stream status inquiries.....ferror(3S)
streamio: STREAMS ioctl commands.....streamio(7)
driver streampty: STREAMS pseudo terminal .....streampty(7)
clone: open any minor device on a STREAMS driver .....clone(7)
strclean: STREAMS error logger cleanup program.....strclean(1M)
strerr: STREAMS error logger daemon .....strerr(1M)
log: interface to STREAMS error logging and event tracing.....log(7)
poll: STREAMS input/output multiplexing.....poll(2)
streamio: STREAMS ioctl commands.....streamio(7)
Transport Interface cooperating STREAMS module.....timod(7)
Interface read/write interface STREAMS module.....tirdwr(7)
sp: STREAMS pipe device.....sp(7)
streampty: STREAMS pseudo terminal driver.....streampty(7)
strace: print STREAMS trace messages.....strace(1M)
streamtty: Generic STREAMS tty driver interface.....streamtty(7)
ld0: STREAMS tty line discipline .....ld0(7)
driver interface streamtty: Generic STREAMS tty .....streamtty(7)
strerr: STREAMS error logger daemon .....strerr(1M)
strftime: format time information.....strftime(3C)
long integer and base-64 ASCII string.....a64l(3C)
tzset: convert date and time to string.....ctime(3C)
convert floating-point number to string.....ecvt(3C)
fgrep: search a file for a character string.....fgrep(1)
regex: match patterns against a string.....regex(1F)
gps: graphical primitive string, format of graphical files.....gps(4)
gets, fgets: get a string from a stream.....gets(3S)
puts, fputs: put a string on a stream.....puts(3S)
echo: put string on virtual output .....echo(1F)
strpbrk, strspn, strcspn, strtok: string operations.....string(3C)

strcmp, strncmp, strcpy, strncpy,/ string: strcat, strdup, strncat, ...string(3C)
strtod, atof: convert string to double-precision number.....strtod(3C)
strtol, atol, atoi: convert string to integer.....strtol(3C)
cftime: language specific strings.....cftime(4)
object file strip: strip an.....see *C and Lang Tools Binder*
/strcmp, strncmp, strcpy, strncpy, strlen, strchr, strrchr, strpbrk,/.....string(3C)
string: strcat, strdup, strncat, strcmp, strncmp, strcpy, strncpy,/.....string(3C)
/strcat, strdup, strncat, strcmp, strncmp, strcpy, strlen,/.....string(3C)
/strncat, strcmp, strncmp, strcpy, strncpy, strlen, strchr, strrchr,/.....string(3C)
/strncpy, strlen, strchr, strrchr, strpbrk, strspn, strcspn, strtok:/.....string(3C)
/strcpy, strncpy, strlen, strchr, strrchr, strpbrk, strspn, strcspn,/.....string(3C)
/strlen, strchr, strrchr, strpbrk, strspn, strcspn, strtok: string/.....string(3C)
character strstr: find first occurrence of.....strstr(3C)
double-precision number strtod, atof: convert string to.....strtod(3C)
strrchr, strpbrk, strspn, strcspn, strtok: string operations.....string(3C)
to integer strtol, atol, atoi: convert string.....strtol(3C)
processes using a file or file structure.....fuser(1M)
getscsimatch: get scsiinfo structure.....getscsiinfo(3SEQ)
getzdinfobyname: get zdinfo structure.....getzdinfo(3SEQ)
read constab entries into a structure.....read_constab(3SEQ)
t_alloc: allocate a library structure.....t_alloc(3N)
t_free: free a library structure.....t_free(3N)
state (defunct) stty, gtty: set and get terminal.....stty(2)
stty: set the options for a terminal.....stty(1)
su: become superuser or another user.....su(1M)
backup.codes1: backup subcode definitions.....backup_codes1(4)
m_fork: execute a subprogram in parallel.....m_fork(3PPS)
dbm_clearerr, dmb_blkflush: database subroutines.....ndbm(3X)
plot: graphics interface subroutines.....plot(3X)
sulog: log of substitute user events.....sulog(4)
sulog: log of substitute user events.....sulog(4)
of a file sum: print checksum and block count.....sum(1)
du: summarize disk usage.....du(1M)
repquota: summarize quotas for a filesystem.....repquota(1M)
records acctcms: command summary from per-process accounting.....acctcms(1M)
s_init_lock, s_lock, s_clock, s_unlock: initialize, lock, unlock locks.....s_lock(3PPS)
sync: update the superblock.....sync(1M)
sync: update superblock.....sync(2)
su: become superuser or another user.....su(1M)
getgroups: get supplementary group IDs.....getgroups(2)
uucp: networking support files.....uucp(4)
lptypes: table of printer types supported by the menu system.....lptypes(4)
m_park_procs, m_rele_procs: suspend and resume child process/.....m_park_procs(3PPS)
sleep: suspend execution for an interval.....sleep(1)
sleep: suspend execution for interval.....sleep(3C)
pause: suspend process until signal.....pause(2)
swab: swap bytes.....swab(3C)
wait at barrier s_init_barrier, s_wait_barrier: initialize barrier,.....s_wait_barrier(3PPS)
swap: swap administrative interface.....swap(1M)
swab: swap bytes.....swab(3C)
swap: swap administrative interface.....swap(1M)
/a command with priority aging, PFF, swapping, or processor migration/.....noage(1M)

seeks on DCC disks sweepon, — sweepoff: enable/disable patrol ..... sweepon(1M)
patrol seeks on DCC disks — sweepon, sweepoff: enable/disable ..... sweepon(1M)
symbol table/ ldgetname: retrieve — symbol name for common object file ..... ldgetname(3X)
symbol name for common object file — symbol table entry ..... ldgetname(3X)
ldtbindex: compute the index of a — symbol table entry of a common/ ..... ldtbindex(3X)
ldtbread: read an indexed — symbol table entry of a common object/ ..... ldtbread(3X)
syms: common object file — symbol table format ..... syms(4)
ldtbseek: seek to the — symbol table of a common object file ..... ldtbseek(3X)
unistd: file header for — symbolic constants ..... unistd(4)
readlink: read value of a — symbolic link ..... readlink(2SEQ)
symlink: make — symbolic link to a file ..... symlink(2SEQ)
of common operating system terms and — symbols ..... glossary(1)
file — symlink: make symbolic link to a ..... symlink(2SEQ)
table format — syms: common object file symbol ..... syms(4)
sync: update superblock ..... sync(2)
sync: update the superblock ..... sync(1M)
controlling FMLI object vsig: — synchronize a co-process with the ..... vsig(1F)
t_sync: — synchronize transport library ..... t_sync(3N)
check documents for correct eqn — syntax ..... checkeq(1)
(command interpreter) with C-like — syntax ..... csh(1)
administration — sysadm: menu interface to do system ..... sysadm(1)
syscall: indirect system call ..... syscall(2)
variables — sysconf: get configurable system ..... sysconf(2)
configuration file — sys_config: format of a system ..... sys_config(4)
perror, errno, — sys_errlist, sys_nerr: system error messages ..... perror(3C)
sysi86: machine specific functions ..... sysi86(2)
menu system functions — syskeymap: identify keys to internal ..... syskeymap(4)
perror, errno, sys_errlist, — sys_nerr: system error messages ..... perror(3C)
multiplexor driver st: — Systech MTI-800/1600/1650 terminal ..... st(7)
lp: — Systech parallel line printer interface ..... lp(7)
bsearch: binary search a sorted — table ..... bsearch(3C)
constab: constructed devices — table ..... constab(4)
devtab: virtual device — table ..... devtab(4)
dltsectbl: refresh security — table ..... dltsectbl(1M)
kernel auto-configuration device — table ..... dumpconf(1M)
fstab: file-system- — table ..... fstab(4)
mnttab: mounted filesystem — table ..... mnttab(4)
setmnt: establish mount — table ..... setmnt(1M)
shells: shell — table ..... shells(4)
task: global security — table ..... task(4)
name for common object file symbol — table entry ..... ldgetname(3X)
/compute the index of a symbol — table entry of a common object file ..... ldtbindex(3X)
ldtbread: read an indexed symbol — table entry of a common object file ..... ldtbread(3X)
syms: common object file symbol — table format ..... syms(4)
tasktbl: create security — table in shared memory ..... tasktbl(1M)
get maximum in-use descriptor — table index ..... getdtablemax(2SEQ)
ldtbseek: seek to the symbol — table of a common object file ..... ldtbseek(3X)
the menu system lptypes: — table of printer types supported by ..... lptypes(4)
getdtablesize: get descriptor — table size ..... getdtablesize(2SEQ)
setdtablesize: set descriptor — table size ..... setdtablesize(2SEQ)
classification and conversion — tables ..... chrtbl(1M)
hdestroy: manage hash search — tables ..... hsearch(3C)

tabs: set — tabs on a terminal ..... tabs(1)
tabs: set tabs on a terminal ..... tabs(1)
t_accept: accept a connect request ..... t_accept(3N)
ctags: create a — tags file ..... ctags(1)
file — tail: deliver the last part of a ..... tail(1)
structure — t_alloc: allocate a library ..... t_alloc(3N)
trig: sin, cos, — tan, asin, acos, atan, atan2: trigonometric/ ..... trig(3M)
sinh, cosh, — tanh: hyperbolic functions ..... sinh(3M)
tm: SCSI — tape controller ..... tm(7)
ts: SCSI — tape controller ..... ts(7)
hpio: Hewlett-Packard 2645A terminal — tape file archiver ..... hpio(1)
tar: — tape file archiver ..... tar(1)
mtio: UNIX magnetic — tape interface ..... mtio(7)
tg: SCSI 1/2-inch magnetic — tape interface ..... tg(7)
xt: Xylogics 472 MULITBUS magnetic — tape interface ..... xt(7)
mt: magnetic — tape manipulating program ..... mt(1)
tar: tape file archiver ..... tar(1)
task: global security table ..... task(4)
shared memory — tasktbl: create security table in ..... tasktbl(1M)
transport endpoint — t_bind: bind an address to a ..... t_bind(3N)
deroff: remove nroff, — tbl, and eqn constructs ..... deroff(1)
terminal state — tcgetattr, tcsetattr: get/set ..... tcgetattr(2)
group ID — tcgetpgrp: get foreground process ..... tcgetpgrp(2)
t_close: close a transport endpoint ..... t_close(3N)
with another transport user — t_connect: establish a connection ..... t_connect(3N)
tcgetattr, — tcsetattr: get/set terminal state ..... tcgetattr(2)
group ID — tcsetpgrp: set foreground process ..... tcsetpgrp(2)
tsearch, tfind, — tdelete, twalk: manage binary search trees ..... tsearch(3C)
tee: pipe fitting ..... tee(1)
4014: paginator for the — Tektronix 4014 terminal ..... 4014(1)
init, — telinit: process control initialization ..... init(1M)
directory: opendir, readdir, — telldir, seekdir, rewinddir,/ ..... directory(3X)
temporary file tmpnam, — tempnam: create a name for a ..... tmpnam(3S)
tmpfile: create a — temporary file ..... tmpfile(3S)
tmpnam, tempnam: create a name for a — temporary file ..... tmpnam(3S)
terminals — term: conventional names for ..... term(5)
term: format of compiled — term file ..... term(4)
term: format of compiled term file ..... term(4)
description captoinfo: convert a — termcap description into a terminfo ..... captoinfo(1M)
base — termcap: terminal capability data ..... termcap(4)
paginator for the Tektronix 4014 — terminal ..... 4014(1)
special functions of the DASI 450 — terminal ..... 450(1)
ct: spawn getty to a remote — terminal ..... ct(1C)
ctermid: generate filename for — terminal ..... ctermid(3S)
hangup the specified controlling — terminal ..... fvhangup(2SEQ)
stty: set the options for a — terminal ..... stty(1)
tabs: set tabs on a — terminal ..... tabs(1)
tty: get the name of the — terminal ..... tty(1)
ttyname, isatty: find name of a — terminal ..... ttyname(3C)
/associated with a terminal or the — terminal associated with a device ..... ttydevname(1)
termcap: — terminal capability data base ..... termcap(4)
terminfo: — terminal capability database ..... terminfo(4)

streampty: STREAMS pseudo terminal driver.....streampty(7)
greek: select terminal filter.....greek(1)
tty: controlling terminal (gentty) interface.....tty(7)
termio: general terminal interface.....termio(7)
dial: establish an out-going terminal line connection.....dial(3C)
st: Systech MTI-800/1600/1650 terminal multiplexor driver.....st(7)
xs: Xylogics 781 terminal multiplexor driver.....xs(7)
tput: initialize a terminal or query terminfo database.....tput(1)
/get the device associated with a terminal or the terminal associated/.....ttydevname(1)
optimization package curses: terminal screen handling and.....curses(3X)
gettydefs: speed and terminal settings used by getty.....gettydefs(4)
tcgetattr, tcsetattr: get/set terminal state.....tcgetattr(2)
stty, gtty: set and get terminal state (defunct).....stty(2)
hpio: Hewlett-Packard 2645A terminal tape file archiver.....hpio(1)
line discipline getty: set terminal type, modes, speed, and.....getty(1M)
line discipline uugetty: set terminal type, modes, speed, and.....uugetty(1M)
functions of DASI 300 and 300s terminals.....300(1)
getpseudotty: get pseudo terminals.....getpseudotty(3SEQ)
special functions of Hewlett-Packard terminals.....hp(1)
term: conventional names for terminals.....term(5)
wait for a child process to stop or terminate.....wait(2)
kill: terminate a process.....kill(1)
exit, _exit: terminate process.....exit(2)
tic: terminfo compiler.....tic(1M)
tput: initialize a terminal or query terminfo database.....tput(1)
convert a termcap description into a terminfo description.....captoinfo(1M)
infocmp: compare or print out terminfo descriptions.....infocmp(1M)
database terminfo: terminal capability.....terminfo(4)
termio: general terminal interface.....termio(7)
of common operating system terms and symbols.....glossary(1)
t_error: produce error message.....t_error(3N)
test: condition evaluation command.....test(1)
isnan: isnand, isnanf: test for floating-point NaN/.....isnan(3C)
ed, red: text editor.....ed(1)
ex: text editor.....ex(1)
edit: text editor (variant of ex for casual users).....edit(1)
errormsg: error message text file.....errormsg(4)
newform: change the format of a text file.....newform(1)
fspec: format specification in text files.....fspec(4)
isismgr: ISIS file manager, updates text files in ISIS file format.....isismgr(1M)
plock: lock process, text, or data in memory.....plock(2)
search trees tsearch, tfind, tdelete, twalk: manage binary.....tsearch(3C)
reporter tfipc: interprocess communication.....tfipc(1M)
login shell tfmenush: assign menu system as user.....tfmenush(1M)
tfnotify: send completion message.....tfnotify(1M)
t_free: free a library structure.....t_free(3N)
interface tg: SCSI 1/2-inch magnetic tape.....tg(7)
service information t_getinfo: get protocol-specific.....t_getinfo(3N)
t_getstate: get the current state.....t_getstate(3N)
merge: three-way file merge.....merge(1)
tic: terminfo compiler.....tic(1M)
batch: execute commands at a later time.....at(1)

view a file one full screen at a time ..... more(1)
setting up an environment at login time ..... profile(4)
stime: set time ..... stime(2)
time: get time ..... time(2)
time, ptime: time a command ..... time(1)
and system activity timex: time a command; report process data ..... timex(1)
acctcon1, acctcon2: connect- time accounting ..... acctcon(1M)
time: get time ..... time(2)
strftime: format time information ..... strftime(3C)
profil: execution time profile ..... profil(2)
time, ptime: time a command ..... time(1)
mktime, tzset: convert date and time to string ..... ctime(3C)
clock: report CPU time used ..... clock(3C)
timezone: set default system time zone ..... timezone(4)
times: get process and child process times ..... times(2)
set file access and modification times times ..... utime(2)
times: get process and child process ..... times(2)
update access and modification times of a file ..... touch(1)
process data and system activity timex: time a command; report ..... timex(1)
zone timezone: set default system time ..... timezone(4)
cooperating STREAMS module timod: Transport Interface ..... timod(7)
read/write interface STREAMS module tirdwr: Transport Interface ..... tirdwr(7)
request t_listen: listen for a connect ..... t_listen(3N)
a transport endpoint t_look: look at the current event on ..... t_look(3N)
tm: SCSI tape controller ..... tm(7)
processor tmp_affinity: bind a process to a ..... tmp_affinity(2SEQ)
tmp_ctl: multi-processor control ..... tmp_ctl(2SEQ)
tmpfile: create a temporary file ..... tmpfile(3S)
temporary file tmpnam, tempnam: create a name for a ..... tmpnam(3S)
/isgraph, isascii, tolower, toupper, toascci, _tolower, _toupper,/ ..... ctype(3C)
/tolower, _toupper, _tolower, toascii: translate characters ..... conv(3C)
conv: toupper, tolower, _toupper, _tolower, toascii: translate/ ..... conv(3C)
/isascii, tolower, toupper, toascci, _tolower, _toupper, setchrclass:/ ..... ctype(3C)
/ispunct, isprint, isgraph, isascii, tolower, toupper, toascci, _tolower,/ ..... ctype(3C)
conv: toupper, tolower, _toupper, _tolower, toascii: translate/ ..... conv(3C)
endpoint t_open: establish a transport ..... t_open(3N)
tsort: topological sort ..... tsort(1)
transport endpoint t_optmgmt: manage options for a ..... t_optmgmt(3N)
acctmerg: merge or add total accounting files ..... acctmerg(1M)
modification times of a file touch: update access and ..... touch(1)
/tolower, toupper, toascci, _tolower, _toupper, setchrclass: character/ ..... ctype(3C)
/isprint, isgraph, isascii, tolower, toupper, toascci, _tolower,/ ..... ctype(3C)
conv: toupper, tolower, _toupper, _tolower, toascii: translate/ ..... conv(3C)
_tolower, toascii: translate/ conv: toupper, tolower, _toupper, ... conv(3C)
terminfo database tput: initialize a terminal or query ..... tput(1)
tr: translate characters ..... tr(1)
ptrace, mptrace: process trace facility ..... ptrace(2)
strace: print STREAMS trace messages ..... strace(1M)
to STREAMS error logging and event tracing ..... log(7)
copy: perform a standalone data transfer ..... copy(8)
init.d: system level transition files directory ..... init.d(4)
_toupper, _tolower, toascii: translate characters ..... conv(3C)

tr: translate characters ..... tr(1)
t_bind: bind an address to a transport endpoint ..... t_bind(3N)
t_close: close a transport endpoint ..... t_close(3N)
look at the current event on a transport endpoint ..... t_look(3N)
t_open: establish a transport endpoint ..... t_open(3N)
t_optmgmt: manage options for a transport endpoint ..... t_optmgmt(3N)
t_unbind: disable a transport endpoint ..... t_unbind(3N)
STREAMS module timod: Transport Interface cooperating ..... timod(7)
interface STREAMS module tirdwr: Transport Interface read/write ..... tirdwr(7)
t_sync: synchronize transport library ..... t_sync(3N)
the scheduler for the uucp file transport program ..... uusched(1M)
uucico: file transport program for the uucp system ..... uucico(1M)
establish a connection with another transport user ..... t_connect(3N)
data sent over a connection t_rcv: receive data or expedited ..... t_rcv(3N)
confirmation from a connect request t_rcvconnect: receive the ..... t_rcvconnect(3N)
disconnect t_rcvdis: retrieve information from ..... t_rcvdis(3N)
orderly release indication t_rcvrel: acknowledge receipt of an ..... t_rcvrel(3N)
t_rcvudata: receive a data unit ..... t_rcvudata(3N)
error indication t_rcvuderr: receive a unit data ..... t_rcvuderr(3N)
ftw: walk a file tree ..... ftw(3C)
tdelete, twalk: manage binary search trees ..... tsearch(3C)
atan, atan2: trigonometric/ trig: sin, cos, tan, asin, acos, ... trig(3M)
cos, tan, asin, acos, atan, atan2: trigonometric functions ..... trig(3M)
true, false: provide truth values ..... true(1)
truncate, ftruncate: truncate a file to a specified length ..... truncate(2SEQ)
to a specified length truncate, ftruncate: truncate a file ..... truncate(2SEQ)
u3b2, u3b5, vax: get processor type truth value ..... machid(1)
true, false: provide truth values ..... true(1)
ts: SCSI tape controller ..... ts(7)
manage binary search trees tsearch, tfind, tdelete, twalk: ..... tsearch(3C)
over a connection t_snd: send data or expedited data ..... t_snd(3N)
disconnect request t_snddis: send user-initiated ..... t_snddis(3N)
release t_sndrel: initiate an orderly ..... t_sndrel(3N)
t_sndudata: send a data unit ..... t_sndudata(3N)
tsort: topological sort ..... tsort(1)
library t_sync: synchronize transport ..... t_sync(3N)
interface tty: controlling terminal (gentty) ..... tty(7)
streamtty: Generic STREAMS tty driver interface ..... streamtty(7)
tty: get the name of the terminal ..... tty(1)
ld0: STREAMS tty line discipline ..... ld0(7)
between major/minor numbers and tty names ..... dev_to_ttyname(3C)
associated with a terminal or the/ ttydevname: get the device ..... ttydevname(1)
terminal ttyname, isatty: find name of a ..... ttyname(3C)
major/minor/ dev_to_ttyname, ttyname_to_dev: convert between ..... dev_to_ttyname(3C)
file of the current user ttyslot: find the slot in the utmp ..... ttyslot(3C)
endpoint t_unbind: disable a transport ..... t_unbind(3N)
parameters vmtune: tune system-wide virtual-memory ..... vmtune(1M)
tunefs: tune up an existing filesystem ..... tunefs(1M)
filesystem tunefs: tune up an existing ..... tunefs(1M)
examine and change virtual memory tuning parameters ..... vm_ctl(2SEQ)
/prdaily, prtacct, shutacct, startup, turnacct: shell procedures for/ ..... acctsh(1M)
tsearch, tfind, tdelete, twalk: manage binary search trees ..... tsearch(3C)

types: primitive system data — types.....types(5)
types: primitive system data types.....types(5)
lptypes: table of printer — types supported by the menu system.....lptypes(4)
/asctime, cftime, ascftime, mktime, — tzset: convert date and time to/.....ctime(3C)
machid: i386, ns32000, pdp11, — u3b, u3b2, u3b5, vax: get processor type/.....machid(1)
machid: i386, ns32000, pdp11, u3b, — u3b2, u3b5, vax: get processor type/.....machid(1)
/i386, ns32000, pdp11, u3b, u3b2, — u3b5, vax: get processor type truth/.....machid(1)
uadmin: administrative control .....uadmin(1M)
uadmin: administrative control.....uadmin(2)
/endspent, fgetspent, lckpwdf, — ulckpwdf: get shadow password file entry.....getspent(3X)
ulimit: get and set user limits .....ulimit(2)
mask — umask: set and get file creation.....umask(2)
umask: set file-creation mode mask.....umask(1)
mount, — umount: mount and unmount filesystems.....mount(1M)
umount: unmount a filesystem .....umount(2)
filesystems mountall, — umountall: mount, unmount multiple.....mountall(1M)
system — uname: get name of current operating.....uname(2)
uname: print name of current system .....uname(1)
expand data compress, — uncompress, zcat: compress and .....compress(1)
unget: — undo a previous get of an SCCS file .....unget(1)
SCCS file — unget: undo a previous get of an .....unget(1)
input stream — ungetc: push character back into .....ungetc(3S)
file — uniq: report repeated lines in a.....uniq(1)
mktemp: make a — unique filename .....mktemp(3C)
constants — unistd: file header for symbolic.....unistd(4)
t_rcvudata: receive a data — unit .....t_rcvudata(3N)
t_sndudata: send a data — unit.....t_sndudata(3N)
t_rcvuderr: receive a — unit data error indication.....t_rcvuderr(3N)
units: conversion program .....units(1)
mtio: — UNIX magnetic tape interface .....mtio(7)
cu: call another — UNIX system .....cu(1C)
link, unlink: link and — unlink files and directories.....link(1M)
link, — unlink: link and unlink files and directories.....link(1M)
unlink: remove directory entry.....unlink(2)
m_lock, m_unlock: lock, — unlock locks .....m_lock(3PPS)
s_clock, s_unlock: initialize, lock, — unlock locks.....s_lock(3PPS)
space munmap: — unmap part of the processs address .....munmap(2SEQ)
spellhist: — unmatched spelling words file.....spellhist(4)
umount: — unmount a filesystem.....umount(2)
mount, umount: mount and — unmount filesystems.....mount(1M)
mountall, umountall: mount, — unmount multiple filesystems.....mountall(1M)
pack, pcat, — unpack: compress and expand files.....pack(1)
set, unset: set and — unset environment variables in core or in files.....set(1F)
variables in core or in files set, — unset: set and unset environment .....set(1F)
lsearch, lfind: linear search and — update.....lsearch(3C)
touch: — update access and modification times of a file.....touch(1)
make: maintain, — update, and regenerate groups of programs .....make(1)
bdflush: — update daemon.....bdflush(1M)
bdflush: — update daemon.....bdflush(2SEQ)
sync: — update superblock.....sync(2)
sync: — update the superblock.....sync(1M)
from/ pwconv: installs and — updates /etc/shadow with information.....pwconv(1M)

isismgr: ISIS file manager, updates text files in ISIS file format ..... isismgr(1M)
been up uptime: show how long system has..... uptime(1)
du: summarize disk usage ..... du(1M)
quota: display disk usage and limits..... quota(1)
retrieve a command description and usage examples ..... usage(1)
description and usage examples usage: retrieve a command..... usage(1)
usclk: 32-bit microsecond counter..... usclk(7)
getusclk, usclk_init: get/init microsecond clock ..... getusclk(3SEQ)
get character login name of the user..... cuserid(3S)
logname: return login name of user ..... logname(3X)
setup: initialize system for first user ..... setup(1)
su: become superuser or another user ..... su(1M)
a connection with another transport user..... t_connect(3N)
slot in the utmp file of the current user..... ttyslot(3C)
write: write to another user ..... write(1)
cpset: copy file and set mode, user, and group IDs..... cpset(1)
setuid, setgid: set user and group IDs..... setuid(2)
id: print user and group IDs and names..... id(1M)
crontab: user crontab file..... crontab(1)
addusr.def: add user defaults..... addusr_def(4)
/geteuid, getgid, getegid: get real user, effective user, real group/..... getuid(2)
environ: user environment ..... environ(5)
run commands performed for multi- user environment..... rc2(1M)
sulog: log of substitute user events ..... sulog(4)
generate disk accounting data by user ID..... diskusg(1M)
ulimit: get and set user limits..... ulimit(2)
tfmenush: assign menu system as user login shell..... tfmenush(1M)
menumsg: route user message to menu message queue..... menumsg(1M)
edquota: edit user quotas..... edquota(1M)
/getegid: get real user, effective user, real group, and effective/..... getuid(2)
t_snddis: send user-initiated disconnect request..... t_snddis(3N)
editor (variant of ex for casual users)..... edit(1)
the operating system for beginning users..... starter(1)
wall: write to all users ..... wall(1)
ut_delete_user: add/delete users in utmp file..... utadduser(3X)
mail, rmail: send mail to users or read mail..... mail(1)
login attempts /usr/adm/loginlog: log of failed/..... loginlog(4)
ustat: get filesystem statistics..... ustat(2)
add/delete users in utmp file ut_add_user, ut_add_user: ..... utadduser(3X)
utmp file ut_delete_user: add/delete users in ..... utadduser(3X)
introduction to system standalone utilities..... intro(8)
get_process_stats: get resource utilization information..... get_process_stats(2SEQ)
modification times utime: set file access and ..... utime(2)
utmp, wtmp: utmp and wtmp entry formats..... utmp(4)
ut_delete_user: add/delete users in utmp file..... utadduser(3X)
endutent, utmpname, getutpid: access utmp file entry..... getut(3C)
ttyslot: find the slot in the utmp file of the current user..... ttyslot(3C)
formats utmp, wtmp: utmp and wtmp entry ..... utmp(4)
/pututline, setutent, endutent, utmpname, getutpid: access utmp file/..... getut(3C)
and permissions file uucheck: check the uucp directories..... uucheck(1M)
the uucp system uucico: file transport program for..... uucico(1M)
clean-up uucleanup: uucp spool directory ..... uucleanup(1M)

uucheck: check the — uucp directories and permissions file.....uucheck(1M)
uusched: the scheduler for the — uucp file transport program.....uusched(1M)
— uucp: networking support files.....uucp(4)
uucleanup: — uucp spool directory clean-up.....uucleanup(1M)
uustat: — uucp status inquiry and job control.....uustat(1C)
file transport program for the — uucp system.....uucico(1M)
UNIX-to-UNIX system/ — uucp, uulog, uuname:.....uucp(1C)
speed, and line discipline — uugetty: set terminal type, modes,.....uugetty(1M)
uucp, — uulog, uuname: UNIX-to-UNIX/.....uucp(1C)
uucp, uulog, — uuname: UNIX-to-UNIX system copy.....uucp(1C)
uuto, — uupick: public UNIX-to-UNIX/.....uuto(1C)
file transport program — uusched: the scheduler for the uucp.....uusched(1M)
control — uustat: uucp status inquiry and job.....uustat(1C)
UNIX-to-UNIX system/ — uuto, uupick: public.....uuto(1C)
with debugging on — Uutry: try to contact remote system.....uutry(1M)
system command execution — uux: UNIX-to-UNIX.....uux(1C)
requests — uuxqt: execute remote command.....uuxqt(1M)
— val: validate SCCS file.....val(1)
val: — validate SCCS file.....val(1)
— valloc: aligned memory allocator.....valloc(3SEQ)
abs: return integer absolute — value.....abs(3C)
u3b5, vax: get processor type truth — value.....machid(1)
getenv: return — value for environment name.....getenv(3C)
floor, ceiling, remainder, absolute — value functions.....floor(3M)
readlink: read — value of a symbolic link.....readlink(2SEQ)
putenv: change or add — value to environment.....putenv(3C)
current form field to its default — values.....reset(1F)
true, false: provide truth — values.....true(1)
values: machine-dependent — values.....values(5)
— values: machine-dependent values.....values(5)
print formatted output of a — varargs argument list.....vprintf(3S)
list — varargs: handle variable argument.....varargs(5)
varargs: handle — variable argument list.....varargs(5)
fpathconf: get configurable pathname — variables.....pathconf(2)
sysconf: get configurable system — variables.....sysconf(2)
.environ, .pref, — .variables: FACE files.....env(4)
unset: set and unset environment — variables in core or in files.....set(1F)
edit: text editor (— variant of ex for casual users).....edit(1)
proc_ctl: control over — various process attributes.....proc_ctl(2SEQ)
/ns32000, pdp11, u3b, u3b2, u3b5, — vax: get processor type truth value.....machid(1)
— vc: version control.....vc(1)
get option letter from argument — vector.....getopt(3C)
readv: read — vector from file.....readv(2SEQ)
writev: write — vector on a file.....writev(2SEQ)
assert: — verify program assertion.....assert(3X)
vc: — version control.....vc(1)
get: get a — version of an SCCS file.....get(1)
sccsdiff: compare two — versions of an SCCS file.....sccsdiff(1)
virtual memory efficient way — vfork: spawn new process in a.....vfork(2)
output of a varargs/ vprintf, — vfprintf, vsprintf: print formatted.....vprintf(3S)
editor based on ex — vi: screen-oriented (visual) display.....vi(1)
more: — view a file one full screen at a time.....more(1)

map physical addresses into processs virtual address space.....pmap(7)
devtab: virtual device table.....devtab(4)
devdestroy: build and destroy virtual devices.....devbuild(1M)
vfork: spawn new process in a virtual memory efficient way.....vfork(2)
vm_ctl: examine and change virtual memory tuning parameters.....vm_ctl(2SEQ)
echo: put string on virtual output.....echo(1F)
controlling terminal fvhangup: virtually hangup the specified.....fvhangup(2SEQ)
vmtune: tune system-wide virtual-memory parameters.....vmtune(1M)
pmap: display or alter parameters of virtual-to-physical mapping devices.....pmap(1M)
vi: screen-oriented visual display editor based on ex.....vi(1)
memory tuning parameters vm_ctl: examine and change virtual.....vm_ctl(2SEQ)
virtual-memory parameters vmtune: tune system-wide.....vmtune(1M)
filesystem volcopy: make literal copy of.....volcopy(1M)
fs, inode: format of file system volume.....fs(4)
formatted output of a varargs/ vprintf, vfprintf, vsprintf: print.....vprintf(3S)
the controlling FMLI object vsig: synchronize a co-process with.....vsig(1F)
a varargs/ vprintf, vfprintf, vsprintf: print formatted output of.....vprintf(3S)
prtvtoc: print the VTOC of a block device.....prtvtoc(1M)
prtvtoc: print the VTOC of a standalone disk device.....prtvtoc(8)
mkvtoc: populate VTOC on disks.....mkvtoc(1M)
vtoc: VTOC partitioning driver.....vtoc(7)
vtoc: VTOC partitioning driver.....vtoc(7)
s_wait_barrier: initialize barrier, wait at barrier.....s_wait_barrier(3PPS)
wait: await completion of process.....wait(1)
wait, waitpid: wait for a child process to stop or terminate.....wait(2)
sigsuspend: wait for a signal.....sigsuspend(2)
process to stop or terminate wait, waitpid: wait for a child.....wait(2)
stop or terminate wait, waitpid: wait for a child process to.....wait(2)
ftw: walk a file tree.....ftw(3C)
wall: write to all users.....wall(1)
wc: word count.....wc(1)
wd: SSM SCSI disk controller.....wd(7)
what: identify SCCS files.....what(1)
whatis: describe what a command is.....whatis(1)
whodo: who is doing what.....whodo(1M)
who: who is on the system.....who(1)
who: who is on the system.....who(1)
whodo: who is doing what.....whodo(1M)
vmtune: tune system- wide virtual-memory parameters.....vmtune(1M)
addformopt: add a form/SELECT window option to a menu.....addformopt(1M)
wc: word count.....wc(1)
fgetc, getw: get character or word from a stream.....getc(3S)
fputc, putw: put character or word on a stream.....putc(3S)
spellhist: unmatched spelling words file.....spellhist(4)
cd: change working directory.....cd(1)
chdir: change working directory.....chdir(2)
getcwd: get pathname of current working directory.....getcwd(3C)
pwd: working directory name.....pwd(1)
specific alarms and/or the working indicator.....indicator(1F)
interface wp: SSM parallel line printer.....wp(7)
lseek: move read/ write file pointer.....lseek(2)
tirdwr: Transport Interface read/ write interface STREAMS module.....tirdwr(7)

write: write on a file ..... write(2)
putpwent: write password file entry ..... putpwent(3C)
putspent: write shadow password file entry ..... putspent(3X)
wall: write to all users ..... wall(1)
write: write to another user ..... write(1)
writev: write vector on a file ..... writev(2SEQ)
write: write on a file ..... write(2)
write: write to another user ..... write(1)
writev: write vector on a file ..... writev(2SEQ)
open: open for reading or writing ..... open(2)
utmp, wtmp: utmp and wtmp entry formats ..... utmp(4)
utmp, wtmp: utmp and wtmp entry formats ..... utmp(4)
accounting records fwtmp, wtmpfix: manipulate connect ..... fwtmp(1M)
and execute command xargs: construct argument list(s) ..... xargs(1)
multiplexor driver xs: Xylogics 781 terminal ..... xs(7)
tape interface xt: Xylogics 472 MULITBUS magnetic ..... xt(7)
xt: Xylogics 472 MULITBUS magnetic tape interface ..... xt(7)
xs: Xylogics 781 terminal multiplexor driver ..... xs(7)
bessel: j0, j1, jn, y0, y1, yn: Bessel functions ..... bessel(3M)
bessel: j0, j1, jn, y0, y1, yn: Bessel functions ..... bessel(3M)
yacc: yet another compiler-compiler ..... yacc(1)
compress, uncompress, zcat: compress and expand data ..... compress(1)
zd: dual-channel disk controller ..... zd(7)
for dual-channel disk controller zdformat: standalone disk formatter ..... zdformat(8)
getzdinfo, getzdinfobyname: get zdinfo structure ..... getzdinfo(3SEQ)

NAME

m4 – macro processor

SYNOPSIS

m4 [options] [files]

DESCRIPTION

The **m4** command is a macro processor intended as a front end for Ratfor, C, and other languages. Each of the argument files is processed in order. If there are no files or if a filename is –, the standard input is read. The processed text is written on the standard output.

The options and their effects are as follows:

–e Operate interactively. Interrupts are ignored and the output is unbuffered.

–s Enable line sync output for the C preprocessor (#line ...)

–B*int* Change the size of the push-back and argument collection buffers from the default of 4096.

–H*int* Change the size of the symbol table hash array from the default of 199. The size should be prime.

–S*int* Change the size of the call stack from the default of 100 slots. Macros take three slots and non-macro arguments take one.

–T*int* Change the size of the token buffer from the default of 512 bytes.

To be effective, these flags must appear before any **–D** or **–U** flags and before any filenames:

–D*name*[=*val*]
Defines *name* to *val* or to null in *val*'s absence.

–U*name*
Undefines *name*.

Macro calls have this form:

name(arg1,arg2, ..., argn)

The (must immediately follow the name of the macro. If the name of a defined macro is not followed by a (, it is deemed to be a call of that macro with no arguments. Potential macro names consist of alphabetic letters, digits, and underscore (_), where the first character is not a digit.

Leading unquoted blanks, tabs, and newlines are ignored while collecting arguments. Left and right single quotes are used to quote strings. The value of a quoted string is the string stripped of the quotes.

When a macro name is recognized, its arguments are collected by searching for a matching right parenthesis. If fewer arguments are supplied than are in the macro definition, the trailing arguments are taken to be null. Macro evaluation proceeds normally during the collection of the arguments, and any commas or right parentheses which happen to turn up within the value of a nested call are as effective as those in the original input text. After argument collection, the value of the macro is pushed back onto the input stream and rescanned.

m4 makes available the following built-in macros. They may be redefined, but once this is done the original meaning is lost. Their values are null unless otherwise stated.

define — The second argument is installed as the value of the macro whose name is the first argument. Each occurrence of $*n* in the replacement text, where *n* is a digit, is replaced by the *n*-th argument. Argument 0 is the name of the macro; missing arguments are replaced by the null string; **$#** is replaced by the number of arguments; **$*** is replaced by a list of all the arguments separated by commas; **$@** is like **$***, but each argument is quoted (with the current quotes).

undefine — Removes the definition of the macro named in its argument.

defn — Returns the quoted definition of its argument(s). It is useful for renaming macros, especially built-ins.

pushdef — Like *define*, but saves any previous definition.

popdef — Removes current definition of its argument(s), exposing the previous one, if any.

ifdef — If the first argument is defined, the value is the second argument, otherwise the third. If there is no third argument, the value is null. The word *unix* is predefined on UNIX system versions of **m4**.

shift — Returns all but its first argument. The other arguments are quoted and pushed back with commas in between. The quoting nullifies the effect of the extra scan that will subsequently be performed.

changequote — Change quote symbols to the first and second arguments. The symbols may be up to five characters long. *Changequote* without arguments restores the original values.

changecom — Change left and right comment markers from the default # and newline. With no arguments, the comment mechanism is effectively disabled. With one argument, the left marker becomes the argument and the right marker becomes newline. With two arguments, both markers are affected. Comment markers may be up to five characters long.

divert — **m4** maintains 10 output streams, numbered 0-9. The final output is the concatenation of the streams in numerical order; initially stream 0 is the current stream. The *divert* macro changes the current output stream to its (digit-string) argument. Output diverted to a stream other than 0 through 9 is discarded.

undivert — Causes immediate output of text from diversions named as arguments, or all diversions if no argument. Text may be undiverted into another diversion. Undiverting discards the diverted text.

divnum — Returns the value of the current output stream.

dnl — Reads and discards characters up to and including the next newline.

ifelse	Has three or more arguments. If the first argument is the same string as the second, then the value is the third argument. If not, and if there are more than four arguments, the process is repeated with arguments 4, 5, 6 and 7. Otherwise, the value is either the fourth string or, if it is not present, null.
incr	Returns the value of its argument incremented by 1. The value of the argument is calculated by interpreting an initial digit-string as a decimal number.
decr	Returns the value of its argument decremented by 1.
eval	Evaluates its argument as an arithmetic expression, using 32-bit arithmetic. Operators include +, –, *, /, %, ^ (exponentiation), bitwise &, |, ^, and ~; relationals; parentheses. Octal and hex numbers may be specified as in C. The second argument specifies the radix for the result; the default is 10. The third argument may be used to specify the minimum number of digits in the result.
len	Returns the number of characters in its argument.
index	Returns the position in its first argument where the second argument begins (zero origin), or –1 if the second argument does not occur.
substr	Returns a substring of its first argument. The second argument is a zero origin number selecting the first character; the third argument indicates the length of the substring. A missing third argument is taken to be large enough to extend to the end of the first string.
translit	Transliterates the characters in its first argument from the set given by the second argument to the set given by the third. No abbreviations are permitted.
include	Returns the contents of the file named in the argument.
sinclude	Is identical to *include*, except that it says nothing if the file is inaccessible.
syscmd	Executes the operating system command given in the first argument. No value is returned.
sysval	The return code from the last call to *syscmd*.
maketemp	Fills in a string of XXXXX in its argument with the current process ID.
m4exit	Causes immediate exit from **m4**. Argument 1, if given, is the exit code; the default is 0.
m4wrap	Argument 1 will be pushed back at final EOF; for example, m4wrap(`cleanup()')
errprint	Prints its argument on the diagnostic output file.
dumpdef	Prints current names and definitions for the named items or for all, if no arguments are given.

traceon — With no arguments, turns on tracing for all macros (including built-ins). Otherwise, turns on tracing for named macros.

traceoff — Turns off trace globally and for any macros specified. Macros specifically traced by *traceon* can be untraced only by specific calls to *traceoff*.

SEE ALSO

cc(1), cpp(1)

NAME

machid: i386, ns32000, pdp11, u3b, u3b2, u3b5, vax – get processor type truth value

SYNOPSIS

i386

ns32000

pdp11

u3b

u3b2

u3b5

vax

DESCRIPTION

The following commands will return a true value (exit code of 0) if you are on the processor the command name indicates.

i386	True if you are on an Intel 386-based computer.
ns32000	True if you are on a National 32000-based computer.
pdp11	True if you are on a PDP-11/45 or PDP-11/70.
u3b	True if you are on a 3B20 computer.
u3b2	True if you are on a 3B2 computer.
u3b5	True if you are on a 3B5 computer.
vax	True if you are on a VAX-11/750 or VAX-11/780.

The commands that do not apply will return a false (non-zero) value. These commands are often used within makefiles (see **make**(1)) and shell procedures (see **sh**(1)) to increase portability.

SEE ALSO

make(1), sh(1), test(1), true(1)

NAME

mail, rmail – send mail to users or read mail

SYNOPSIS

Sending mail:

mail [**–wt**] persons

rmail [**–wt**] persons

Reading mail:

mail [**–ehpqr**] [**–f** file] [**–F** persons]

DESCRIPTION

Sending mail:

The following command-line arguments affect sending mail:

–w Causes a letter to be sent to a remote user without waiting for the completion of the remote transfer program.

–t Causes a **To:** line showing the intended recipients to be added to the letter.

A *person* is usually a user name recognized by **login**(1). When *persons* are named, **mail** assumes a message is being sent (except in the case of the **–F** option). It reads from the standard input up to an end-of-file (Ctrl-D), or until it reads a line consisting of just a period. When either of those signals is received, **mail** adds the letter to the *mailfile* for each *person*. The *mailfile* of the *person* must be writeable by group 'mail', or the letter will be undeliverable. A letter is a message preceded by the sender's name and a postmark. A postmark consists of one or more **From** lines followed by a blank line.

If a letter is found to be undeliverable, it is returned to the sender with diagnostics that indicate the location and nature of the failure. If **mail** is interrupted during input, the file *dead.letter* is saved to allow editing and resending. *dead.letter* is recreated every time it is needed, erasing any previous contents.

rmail permits only the sending of mail; **uucp**(1C) uses **rmail** as a security precaution.

If the local system has the Basic Networking Utilities installed, mail may be sent to a recipient on a remote system. Prefix *person* by the system name and an exclamation point. A series of system names separated by exclamation points can be used to direct a letter through an extended network.

Reading Mail:

The following command-line arguments affect reading mail:

–e Causes mail not to print. An exit value of 0 is returned if the user has mail. Otherwise, an exit value of 1 is returned.

–h Causes a window of headers to be displayed rather than the latest message. The display is followed by the **?** prompt.

–p Causes all messages to print without prompting for disposition.

–q Causes **mail** to terminate after interrupts. Normally an interrupt causes only the termination of the message being printed.

–r Causes messages to be printed in first-in, first-out order.

–f*file* Causes **mail** to use *file* (for example, *mbox*) instead of the default *mailfile*.

–F*persons* Entered into an empty *mailbox*, causes all incoming mail to be forwarded to *persons*.

mail, unless otherwise influenced by command-line arguments, prints a user's mail messages in last-in, first-out order. For each message, the user is prompted with a **?** and a line is read from the standard input. The following commands are available to determine the disposition of the message:

<newline>, **+**, or **n** Go to the next message.

d, or **dp** Delete message and go to the next message.

d # Delete message number #. Do not go to the next message.

dq Delete message and quit **mail**.

h Display a window of headers around current message.

h # Display header of message number #.

h a Display headers of ALL messages in the user's *mailfile*.

h d Display headers of messages scheduled for deletion.

p Reprint current message.

– Print previous message.

a Print message that arrived during the **mail** session.

Print message number #.

r [*users*] Reply to the sender and other *user(s)*, then delete the message.

s [*files*] Save message in the named *files* (*mbox* is default).

y Same as save.

u [**#**] Undelete message number # (default is last read).

w [*files*] Save message, without its top-most header, in the named *files* (*mbox* is default).

m [*persons*] Mail the message to the named *persons*.

q, or **Ctrl-D** Put undeleted mail back in the *mailfile* and quit **mail**.

x Put all mail back in the *mailfile* unchanged and exit **mail**.

!*command* Escape to the shell to execute *command*.

? Print a command summary.

When a user logs in, the presence of mail (if any) is indicated. Also, notification is made if new mail arrives while using **mail**.

The *mailfile* may be manipulated in two ways to alter the function of **mail**. The *other* permissions of the file may be read-write, read-only, or neither read nor write to allow different levels of privacy. If changed to other than the default, the file will be preserved even when empty to perpetuate the desired permissions.

The file can also contain this first line:

`Forward to` *person*

This line causes all mail sent to the owner of the *mailfile* to be forwarded to *person*. A `Forwarded by...` message is added to the header. This is especially useful in a multi-machine environment to forward all of a person's mail to a single machine and to keep the recipient informed if the mail has been forwarded. Installation and removal of forwarding is done with the **-F** option.

To forward all of one's mail to *systema!user*, enter this command:

mail -Fsystema!user

To forward to more than one user, enter this command:

mail -F"user1,systema!user2,systema!systemb!user3"

Note that when more than one user is specified, the whole list should be enclosed in double quotes so that it can be interpreted as the operand of the **-F** option. The list can be up to 1024 bytes; either commas or white space can be used to separate users.

The following list of characters should not be used anywhere in the **mail -F** argument list or in the "Forward to" line:

; & | ^ , < > ` ( ) <CR>

To remove forwarding, enter this command:

mail -F ""

The pair of double quotes is mandatory to set a NULL argument for the **-F** option.

For forwarding to work properly, the *mailfile* should have "mail" as the group ID and the group permission should be read-write.

FILES

/etc/passwd	To identify sender and locate persons
/usr/mail/*user*	Incoming mail for *user* (the *mailfile*)
$HOME/mbox	Saved mail
$MAIL	Variable containing pathname of *mailfile*
/tmp/ma*	Temporary file
/usr/mail/*.lock	Lock for mail directory
dead.letter	Unmailable text

SEE ALSO

login(1), mailx(1), write(1)

WARNING

The "Forward to person" feature may result in a loop if *sys1!userb* forwards to *sys2!userb* and *sys2!userb* forwards to *sys1!userb*. The symptom is a message saying `unbounded...saved mail in dead.letter`.

BUGS

Conditions sometimes result in a failure to remove a lock file.

After an interrupt, the next message might not print; printing can be forced by typing a **p**.

NAME

mailbug – submit a bug report or enhancement request

SYNOPSIS

mailbug

DESCRIPTION

The **mailbug** utility allows users to submit machine-readable bug reports to a Sequent customer service representative. This service is useful only if the customer service representative supports this method of problem tracking. Check with your system administrator to ensure that the **mailbug** service is supported before submitting a bug report.

mailbug prompts for the information needed to fill out the bug report. After the user has typed in all the required information, **mailbug** allows the user to inspect and edit the bug report before it is mailed, or to cancel the report completely. Unless the report is canceled, **mailbug** sends a copy to the customer service representative and a copy to the user submitting the report. Refer to your system administration documentation for information on installing the **mailbug** service.

mailbug asks for the following information:

Your company or organization name, unless the file */usr/lib/mailbug_co_name* exists. */usr/lib/mailbug_co_name* is a text file that might look like this:

```
Sequent Computer Systems, Inc
Department: New Product Introduction
```

Your full name and phone number, unless the file *.mailbug* exists in your home directory. *.mailbug* is a text file that might look like this:

```
Anne Harris
Phone Number: 123-4567
```

A one-line summary of the bug.

A full description of the bug, which may be as long as necessary. (See below for "~" commands which are available).

Whether the problem happens on the system you are running **mailbug** on.

An assessment of the severity of the bug. The choices are:

Critical — an essential component of the hardware or operating system cannot be used.

Serious — a non-essential module or utility cannot be used.

Normal — some feature of a non-essential module or utility doesn't work correctly, and there is no convenient workaround.

Low — some feature of a non-essential module or utility doesn't work correctly, but a user can work around the problem without serious inconvenience.

Enhancement — suggestions on how the product can be improved.

For Your Information — for information only, no action need be taken.

A guess as to the source of the problem: documentation, hardware, mechanical, software, or other.

For documentation problems, **mailbug** prompts for the name of the manual involved, the part number of the manual from the title page, and the page numbers.

For hardware and mechanical bugs, **mailbug** prompts for the type of hardware component (e.g., MULTIBUS disk controller board, peripheral mounting tower), and its part number, revision number, and serial number.

For software bugs, **mailbug** prompts for the pathname of the file(s) that the user suspects are incorrect — e.g., */usr/ucb/man*.

Once the information is typed in, the user is given the opportunity to display the mailbug, edit it, abort (don't send the mailbug), or submit the mailbug.

If the requested information is unknown or is not applicable, some fields can be skipped by simply typing Return in response to the prompt.

While the problem description is being typed in, certain operations can be performed on the input:

If a line in the description is "~v", then the user's **$EDITOR** (or **vi** by default) is started to edit the description text typed in thus far.

If a line in the description is of the form "~r *filename*", then the contents of *filename* are included in the text of the description.

SEE ALSO

uucp(1)

NAME

mailx – interactive message processing system

SYNOPSIS

mailx [*options*] [*name...*]

DESCRIPTION

The command **mailx** provides a comfortable, flexible environment for sending and receiving messages electronically. When you are reading mail, **mailx** provides commands to facilitate saving, deleting, and responding to messages. When you are sending mail, **mailx** allows editing, reviewing and other modification of the message as it is entered.

For each user, incoming mail is stored in a standard file called the *mailbox*. When **mailx** is called to read messages, the *mailbox* is the default place to find them. As messages are read, they are marked to be moved to a secondary file for storage unless specific action is taken so that the messages need not be seen again. This secondary file is called the *mbox* and is normally located in the user's home directory (see **MBOX (Environment Variables)** for a description of this file). Messages can be saved in other secondary files named by the user. Messages remain in a secondary file until forcibly removed.

The user can access a secondary file by using the **–f** option of the **mailx** command. Messages in the secondary file can then be read or otherwise processed by using the same commands as in the primary *mailbox*. This gives rise within these pages to the notion of a current *mailbox*.

On the command line, *options* start with a dash (–) and any other arguments are taken to be destinations (recipients). If no recipients are specified, **mailx** will attempt to read messages from the *mailbox*. Command line options are as follows:

–e	Test for presence of mail. *mailx* prints nothing and exits with a successful return code if there is mail to read.
–f [*filename*]	Read messages from *filename* instead of *mailbox*. If no *filename* is specified, the *mbox* is used.
–F	Record the message in a file named after the first recipient. Overrides the **record** variable, if set (see **Environment Variables**).
–h *number*	The number of network hops made so far. This is provided for network software to avoid infinite delivery loops. (See **addsopt** under **Environment Variables**).
–H	Print header summary only.
–i	Ignore interrupts. See also **ignore (Environment Variables**).
–n	Do not initialize from the system default *mailx.rc* file.
–N	Do not print initial header summary.
–r *address*	Pass *address* to network delivery software. All tilde commands are disabled. (See **addsopt** under **Environment Variables**).
–s *subject*	Set the Subject header field to *subject*.

-u *user* Read *user's mailbox*. This is only effective if *user's mailbox* is not read protected. All *user's mailbox* must be readable by group mail.

-U Convert **uucp** style addresses to internet standards. Overrides the **conv** environment variable. (See **addsopt** under **Environment Variables**)

When you are reading mail, **mailx** is in command mode. A header summary of the first several messages is displayed, followed by a prompt indicating **mailx** can accept regular commands (see **Commands** below). When you are sending mail, **mailx** is in input mode. If no subject is specified on the command line, a prompt for the subject is printed. (A subject longer than 1024 characters will cause **mailx** to dump core). As the message is typed, **mailx** will read the message and store it in a temporary file. Commands may be entered by beginning a line with the tilde (~) escape character followed by a single command letter and optional arguments. See **Tilde Escapes** for a summary of these commands.

At any time, the behavior of **mailx** is governed by a set of environment variables. These are flags and valued parameters which are set and cleared via the **set** and **unset** commands. See **Environment Variables** below for a summary of these parameters.

Recipients listed on the command line may be of three types: login names, shell commands, or alias groups. Login names may be any network address, including mixed network addressing. If mail is found to to undeliverable, an attempt is made to return it to the sender's *mailbox*. If the recipient name begins with a pipe symbol (|), the rest of the name is taken to be a shell command to pipe the message through. This provides an automatic interface with any program that reads the standard input, such as **lp**(1) for recording outgoing mail on paper. Alias groups are set by the **alias** command (see **Commands** below) and are lists of recipients of any type.

Regular commands have the following form:

[**command**] [*msglist*] [*arguments*]

If no command is specified in *command mode*, **print** is assumed. In *input mode*, commands are recognized by the escape character, and lines not treated as commands are taken as input for the message.

Each message is assigned a sequential number. There is at any time the notion of a current message, marked by a right angle bracket (>) in the header summary. Many commands take an optional list of messages (*msglist*) to operate on. The default for *msglist* is the current message. A *msglist* is a list of message identifiers, separated by spaces, that can include the following:

n Message number *n*

. The current message

^ The first undeleted message

$ The last message

***** All messages

n–m An inclusive range of message numbers

user All messages from **user**

/string All messages with **string** in the subject line (case ignored)

:c All messages of type *c*, where *c* is one of the following:

d Deleted messages
n New messages
o Old messages
r Read messages
u Unread messages

Note that the context of the command determines whether this type of message specification makes sense.

Other arguments are usually arbitrary strings whose usage depends on the command involved. Filenames, where expected, are expanded via the normal shell conventions. Special characters are recognized by certain commands and are documented with the commands below.

At start-up time, **mailx** tries to execute commands from the optional system-wide file (*/usr/lib/mailx/mailx.rc*) to initialize certain parameters, then from a private start-up file (*$HOME/.mailrc*) for personalized variables. With the exceptions noted below, regular commands are legal inside start-up files. The most common use of a start-up file is to set up initial display options and alias lists. The following commands are not legal in the start-up file: **!**, **Copy**, **edit**, **followup**, **Followup**, **hold**, **mail**, **preserve**, **reply**, **Reply**, **shell**, and **visual**. An error in the start-up file causes the remaining lines in the file to be ignored. The *.mailrc* file is optional, and must be constructed locally.

Commands

The following is a complete list of **mailx** commands:

!*shell-command*
Escape to the shell. See **SHELL (Environment Variables)**.

#*comment*
Null command (comment). This may be useful in *.mailrc* files.

= Print the current message number.

? Print a summary of commands.

alias *alias name ...*
group *alias name ...*
Declare an alias for the given names. The names will be substituted when *alias* is used as a recipient. Useful in the *.mailrc* file.

alternates *name ...*
Declares a list of alternate names for your login. When responding to a message, these names are removed from the list of recipients for the response. With no arguments, **alternates** prints the current list of alternate names. See also **allnet (Environment Variables)**.

cd [*directory*]
chdir [*directory*]
Change directory. If *directory* is not specified, **$HOME** is used.

copy [*filename*]
copy [*msglist*] *filename*
Copy messages to the file without marking the messages as saved. Otherwise equivalent to the **save** command.

Copy [*msglist*]
Save the specified messages in a file whose name is derived from the author of the message to be saved, without marking the messages as saved. Otherwise equivalent to the **Save** command.

delete [*msglist*]
Delete messages from the *mailbox*. If **autoprint** is set, the message following the last deleted message prints (see **Environment Variables**).

discard [*header-field* ...]
ignore [*header-field* ...]
Suppresses printing of the specified header fields when displaying messages on the screen. Examples of header fields to ignore are *status* and *cc*. The fields are included when the message is saved. The **Print** and **Type** commands override this command.

dp [*msglist*]
dt [*msglist*]
Delete the specified messages from the *mailbox* and print the message following the last deleted message. Roughly equivalent to a **delete** command followed by a **print** command.

echo *string* ...
Echo the given strings (like **echo**(1)).

edit [*msglist*]
Edit the given messages. The messages are placed in a temporary file and the **EDITOR** variable is used to get the name of the editor (see **Environment Variables**). Default editor is **ed**(1).

exit
xit Exit from **mailx** without changing the *mailbox*. No messages are saved in the *mbox* (see also **quit**).

file [*filename*]
folder [*filename*]
Quit from the current file of messages and read in the specified file. Several special characters are recognized when used as filenames with the following substitutions:

%	The current *mailbox*
%user	The *mailbox* for **user**
#	The previous file
&	The current *mbox*

Default file is the current *mailbox*.

folders
Print the names of the files in the directory set by the **folder** variable (see **Environment Variables**).

followup [*message*]
Respond to a message, recording the response in a file whose name is

derived from the author of the message. Overrides the **record** variable, if set. See also the **Followup**, **Save**, and **Copy** commands and **outfolder (Environment Variables)**.

Followup [*msglist*]
Respond to the first message in the *msglist*, sending the message to the author of each message in the *msglist*. The subject line is taken from the first message and the response is recorded in a file whose name is derived from the author of the first message. See also the **followup**, **Save**, and **Copy** commands and **outfolder (Environment Variables)**.

from [*msglist*]
Prints the header summary for the specified messages.

group *alias name ...*
alias *alias name ...*
Declare an alias for the given names. The names will be substituted when *alias* is used as a recipient. Useful in the *.mailrc* file.

headers [*message*]
Prints the page of headers which includes the message specified. The **screen** variable sets the number of headers per page (see **Environment Variables**). See also the **z** command.

help Prints a summary of commands.

hold [*msglist*]
preserve [*msglist*]
Holds the specified messages in the *mailbox*.

if *s* | *r*
mail-commands
else
mail-commands
endif
Conditional execution, where **s** will execute following *mail-commands*, up to an **else** or **endif**, if the program is in send mode, and **r** causes the *mail-commands* to be executed only in receive mode. Useful in the *.mailrc* file.

ignore *header-field ...*
discard *header-field ...*
Suppresses printing of the specified header fields when displaying messages on the screen. Examples of header fields to ignore are *status* and *cc*. All fields are included when the message is saved. The **Print** and **Type** commands override this command.

list Prints all commands available. No explanation is given.

mail *name ...*
Mail a message to the specified users.

Mail *name*
Mail a message to the specified user and record a copy of it in a file named after that user.

mbox [*msglist*]
Arrange for the given messages to end up in the standard *mbox* save file when **mailx** terminates normally. See **MBOX (Environment Variables)** for a description of this file. See also the **exit** and **quit** commands.

next [*message*]
Go to next message matching *message*. A *msglist* may be specified, but in this case the first valid message in the list is the only one used. This is useful for jumping to the next message from a specific user, since the name would be taken as a command in the absence of a real command. See the discussion of *msglist*s above for a description of possible message specifications.

pipe [*msglist*] [*shell-command*]
| "" [*msglist*] [*shell-command*]
Pipe the message through the given *shell-command*. The message is treated as if it were read. If no arguments are given, the current message is piped through the command specified by the value of the **cmd** variable. If the **page** variable is set, a form feed character is inserted after each message (see **Environment Variables**).

preserve [*msglist*]
hold [*msglist*]
Preserve the specified messages in the *mailbox*.

Print [*msglist*]
Type [*msglist*]
Print the specified messages on the screen, including all header fields. Overrides suppression of fields by the **ignore** command.

print [*msglist*]
type [*msglist*]
Print the specified messages. If **crt** is set, the messages longer than the number of lines specified by the **crt** variable are paged through the command specified by the **PAGER** variable. The default command is **pg**(1) (see **Environment Variables**).

quit Exit from **mailx**, storing messages that were read in *mbox* and unread messages in the *mailbox*. Messages that have been explicitly saved in a file are deleted.

Reply [*msglist*]
Respond [*msglist*]
Send a response to the author of each message in the *msglist*. The subject line is taken from the first message. If **record** is set to a filename, the response is saved at the end of that file (see **Environment Variables**).

reply [*message*]
respond [*message*]
Reply to the specified message, including all other recipients of the message. If **record** is set to a filename, the response is saved at the end of that file (see **Environment Variables**).

Save [*msglist*]
Save the specified messages in a file whose name is derived from the author of the first message. The name of the file is the author's name with all network addressing stripped off. See the **Copy**, **followup**, and **Followup** commands and **outfolder** (**Environment Variables**).

save [*filename*]
save [*msglist*] *filename*
Save the specified messages in the given file. The file is created if it does not exist. The message is deleted from the *mailbox* when **mailx** terminates unless **keepsave** is set (see also **Environment Variables** and the **exit** and **quit** commands).

set
set *name*
set *name=string*
set *name=number*
Define a variable called *name*. The variable may be given a null, string, or numeric value. **set** by itself prints all defined variables and their values. See **Environment Variables** for detailed descriptions of the **mailx** variables.

shell Invoke an interactive shell (see **SHELL** (**Environment Variables**)).

size [*msglist*]
Print the size in characters of the specified messages.

source *filename*
Read commands from the given file and return to command mode.

top [*msglist*]
Print the top few lines of the specified messages. If the **toplines** variable is set, it is taken as the number of lines to print (see **Environment Variables**). The default is 5.

touch [*msglist*]
Touch the specified messages. If any message in *msglist* is not specifically saved in a file, it will be placed in the *mbox* (or the file specified in the **MBOX** environment variable) upon normal termination. See **exit** and **quit**.

Type [*msglist*]
Print [*msglist*]
Print the specified messages on the screen, including all header fields. Overrides suppression of fields by the **ignore** command.

type [*msglist*]
print [*msglist*]
Print the specified messages. If **crt** is set, the messages longer than the number of lines specified by the **crt** variable are paged through the command specified by the **PAGER** variable. The default command is **pg**(1) (see **Environment Variables**).

undelete [*msglist*]
Restore the specified deleted messages. Will only restore messages deleted in the current mail session. If **autoprint** is set, the last message of those restored is printed (see **Environment Variables**).

unset *name ...*
Causes the specified variables to be erased. If the variable was imported from the execution environment (i.e., a shell variable) then it cannot be erased.

version
Prints the current version and release date.

visual [*msglist*]
Edit the given messages with a screen editor. The messages are placed in a temporary file and the **VISUAL** variable is used to get the name of the editor (see **Environment Variables**).

write [*msglist*] *filename*
Write the given messages on the specified file, minus the header and trailing blank line. Otherwise equivalent to the save command.

xit
exit Exit from **mailx** without changing the *mailbox*. No messages are saved in the *mbox* (see also **quit**).

z[+ | –]
Scroll the header display forward or backward one screen. The number of headers displayed is set by the **screen** variable (see **Environment Variables**).

Tilde Escapes

The following commands can be entered only from input mode. You must begin the line with the tilde escape character (~). See **escape** (**Environment Variables**) for directions on changing this special character.

~! *shell-command*
Escape to the shell.

~. Simulate end of file (terminate message input).

~: *mail-command*
~_ *mail-command*
Perform the command-level request. Valid only when sending a message while reading mail.

~? Print a summary of tilde escapes.

~A Insert the autograph string **Sign** into the message (see **Environment Variables**).

~a Insert the autograph string **sign** into the message (see **Environment Variables**).

~b *name ...*
Add the *names* to the blind carbon copy (Bcc) list.

~c *name ...*
Add the *names* to the carbon copy (Cc) list.

~d Read in the *dead.letter* file. See **DEAD** (**Environment Variables**) for a description of this file.

~e Invoke the editor on the partial message. See also **EDITOR** (**Environment Variables**).

~f [*msglist*]
Forward the specified messages. The messages are inserted into the message without alteration.

~h Prompt for *Subject* line and *To*, *Cc*, and *Bcc* lists. If the field has an initial value, it can be edited as if you just typed it.

~i *string*
Insert the value of the named variable into the text of the message. For example, **~A** is equivalent to **~i Sign**. Environment variables set and exported in the shell are also accessible by **~i**.

~m[*msglist*]
Insert the specified messages into the letter, shifting the new text to the right one tab stop. Valid only when sending a message while reading mail.

~p Print the message being entered.

~q Quit from input mode by simulating an interrupt. If the body of the message is not null, the partial message is saved in *dead.letter*. See **DEAD (Environment Variables)** for a description of this file.

~r *filename*
~< *filename*
~< **!***shell-command*
Read in the specified file. If the argument begins with an exclamation point (!), the rest of the string is taken as an arbitrary shell command and is executed, with the standard output inserted into the message.

~s *string* ...
Set the subject line to *string*.

~t *name* ...
Add the given *name*s to the To list.

~v Invoke a preferred screen editor on the partial message. See also **VISUAL (Environment Variables)**.

~w *filename*
Write the partial message onto the given file, without the header.

~x Exit as with **~q** except the message is not saved in *dead.letter*.

~! *shell-command*
Pipe the body of the message through the given *shell-command*. If the *shell-command* returns a successful exit status, the output of the command replaces the message.

Environment Variables

The following environment variables are taken from the execution environment and are not alterable within **mailx**.

HOME=*directory*
The user's base of operations.

MAILRC=*filename*
The name of the start-up file. Default is *$HOME/.mailrc*.

The following variables are internal **mailx** variables. They may be imported from the execution environment or set via the **set** command at any time. The **unset** command may be used to erase variables.

addsopt Enabled by default. If **/bin/mail** is not being used as the deliverer, **noaddsopt** should be specified. (See **WARNINGS** below.)

allnet All network names whose last component (login name) match are treated as identical. This causes the *msglist* message specifications to behave similarly. Default is **noallnet**. See also the **alternates** command and the **metoo** variable.

append Upon termination, append messages to the end of the *mbox* file instead of prepending them. Default is **noappend.**

askcc Prompt for the *Cc* list after message is entered. Default is **noaskcc.**

asksub Prompt for subject if it is not specified on the command line with the **–s** option. Enabled by default.

autoprint
Enable automatic printing of messages after **delete** and **undelete** commands. Default is **noautoprint.**

bang Enable the special-casing of exclamation points (!) in shell escape command lines as in **vi**(1). Default is **nobang.**

cmd=*shell-command*
Set the default command for the **pipe** command. No default value.

conv=*conversion*
Convert **uucp** addresses to the specified address style. The only valid conversion now is *internet*, which requires a mail delivery program conforming to the RFC822 standard for electronic mail addressing. Conversion is disabled by default. See also **sendmail** and the **–U** command line option.

crt=*number*
Pipe messages having more than *number* lines through the command specified by the value of the **PAGER** variable (**pg**(1) by default). Disabled by default.

DEAD=*filename*
The name of the file in which to save partial letters in case of untimely interrupt. Default is *$HOME/dead.letter*.

debug Enable verbose diagnostics for debugging. Messages are not delivered. Default is **nodebug.**

dot Take a period on a line by itself during input from a terminal as end-of-file. Default is **nodot.**

EDITOR=*shell-command*
The command to run when the **edit** or **~e** command is used. Default is **ed**(1).

escape=*c* Substitute *c* for the **~** escape character. Takes effect with next message sent.

folder=*directory*
The directory for saving standard mail files. User-specified filenames beginning with a plus (+) are expanded by preceding the filename with this directory name to obtain the real filename. If *directory* does not start with a slash (/), **$HOME** is prepended to it. In order to use the plus (+) construct on a **mailx** command line, **folder** must be an exported **sh** environment variable. There is no default for the **folder** variable. See also **outfolder** below.

header Enable printing of the header summary when entering **mailx**. Enabled by default.

hold Preserve all messages that are read in the *mailbox* instead of putting them in the standard *mbox* save file. Default is **nohold**.

ignore Ignore interrupts while entering messages. Handy for noisy dial-up lines. Default is **noignore**.

ignoreeof
Ignore end-of-file during message input. Input must be terminated by a period (.) on a line by itself or by the ~. command. Default is **noignoreeof**. See also **dot** above.

keep When the *mailbox* is empty, truncate it to zero length instead of removing it. Disabled by default.

keepsave
Keep messages that have been saved in other files in the *mailbox* instead of deleting them. Default is **nokeepsave**.

MBOX=*filename*
The name of the file used to save messages that have been read. The **x**it command overrides this function, as does saving the message explicitly in another file. Default is *$HOME/mbox*.

metoo If your login appears as a recipient, do not delete it from the list. Default is **nometoo**.

LISTER=*shell-command*
The command (and options) to use when listing the contents of the **folder** directory. The default is **ls**(1).

onehop When responding to a message that was originally sent to several recipients, the other recipient addresses are normally forced to be relative to the originating author's machine for the response. This flag disables alteration of the recipients' addresses, improving efficiency in a network where all machines can send directly to all other machines (that is, one hop away).

outfolder
Causes the files used to record outgoing messages to be located in the directory specified by the **folder** variable unless the pathname is absolute. Default is **nooutfolder**. See **folder** above and the **S**ave, **C**opy, **f**ollowup, and **F**ollowup commands.

page Used with the **p**ipe command to insert a form feed after each message sent through the pipe. Default is **nopage**.

PAGER=*shell-command*
The command to use as a filter for paginating output. This can also be used to specify the options to be used. Default is **pg**(1).

prompt=*string*
Set the *command mode* prompt to *string*. Default is "**?** ".

quiet Refrain from printing the opening message and version when entering **mailx**. Default is **noquiet**.

record=*filename*
Record all outgoing mail in *filename*. Disabled by default. See also **outfolder** above.

save Enable saving of messages in *dead.letter* on interrupt or delivery error. See **DEAD** for a description of this file. Enabled by default.

screen=*number*
Sets the number of lines in a screen of headers for the **headers** command.

sendmail=*shell-command*
Alternate command for delivering messages. Default is **/bin/rmail**(1).

sendwait Wait for background mailer to finish before returning. Default is **nosendwait**.

SHELL=*shell-command*
The name of a preferred command interpreter. Default is **sh**(1).

showto When displaying the header summary and the message is from you, print the recipient's name instead of the author's name.

sign=*string*
The variable inserted into the text of a message when the **~a** (autograph) command is given. There is no default value. (See **~i (Tilde Escapes)**).

Sign=*string*
The variable inserted into the text of a message when the **~A** command is given. There is no default. (See **~i (Tilde Escapes)**).

toplines=*number*
The number of lines of header to print with the **top** command. Default is 5.

VISUAL=*shell-command*
The name of a preferred screen editor. Default is **vi**(1).

FILES

$HOME/.mailrc	Personal start-up file
$HOME/mbox	Secondary storage file
/usr/mail/*	Post office directory
/usr/lib/mailx/mailx.help*	Help message files
/usr/lib/mailx/mailx.rc	Optional global start-up file
/tmp/R[emqsx]*	Temporary files

SEE ALSO

ls(1), mail(1), pg(1)

WARNINGS

The **-h**, **-r** and **-U** options can be used only if **mailx** is built with a delivery program other than **/bin/mail**.

BUGS

Where *shell-command* is shown as valid, arguments are not always allowed. Experimentation is recommended.

Internal variables imported from the execution environment cannot be **unset**.

The full internet addressing is not fully supported by **mailx**. The new standards need some time to settle down.

Attempts to send a message having a line consisting only of a "." are treated as the end of the message by **mail**(1) (the standard mail delivery program).

NAME

make – maintain, update, and regenerate groups of programs

SYNOPSIS

make [**–f** makefile] [**–Pn**] [**–p**] [**–i**] [**–k**] [**–s**] [**–r**] [**–n**] [**–b**] [**–e**] [**–t**] [**–q**] [names]

DESCRIPTION

The **make** command allows the programmer to maintain, update, and regenerate groups of computer programs. The following is a brief description of all options and some special names:

–f *makefile* Description filename. *makefile* is assumed to be the name of a description file.

–p Print out the complete set of macro definitions and target descriptions.

–P*n* Permit *n* command sequences to be done in parallel with '&'. See **Parallel Support** section below. If this option is present multiple times on the command line, the least value for *n* supercedes the others. If this option is missing, **make** looks for the environment variable **PARALLEL** and uses its numeric value for *n*. If **PARALLEL** is undefined, *n* defaults to 3.

–i Ignore error codes returned by invoked commands. This mode is entered if the fake target name **.IGNORE** appears in the description file.

–k Abandon work on the current entry if it fails, but continue on other branches that do not depend on that entry.

–s Silent mode. Do not print command lines before executing. This mode is also entered if the fake target name **.SILENT** appears in the description file.

–r Do not use the built-in rules.

–n No execute mode. Print commands, but do not execute them. Even lines beginning with an at sign (@) are printed.

–b Compatibility mode for old makefiles.

–e Environment variables override assignments within makefiles.

–t Touch the target files (causing them to be up-to-date) rather than issue the usual commands.

–q Question. The **make** command returns a zero or non-zero status code depending on whether the target file is or is not up-to-date.

.DEFAULT If a file must be made but there are no explicit commands or relevant built-in rules, the commands associated with the name **.DEFAULT** are used if it exists.

.PRECIOUS Dependents of this target are not removed when quit or interrupt are hit.

.SILENT Same effect as the **–s** option.

.IGNORE Same effect as the **–i** option.

make executes commands in *makefile* to update one or more target *names*. *name* is typically a program. If no **–f** option is present, *makefile*, *Makefile*, and the Source Code Control System (SCCS) files *s.makefile*, and *s.Makefile* are tried in order. If *makefile* is –, the standard input is taken. More than one **–f** *makefile* argument pair may appear.

make updates a target only if its dependents are newer than the target. All prerequisite files of a target are added recursively to the list of targets. Missing files are deemed to be out-of-date.

makefile contains a sequence of entries that specify dependencies. The first line of an entry is a blank-separated, non-null list of targets, then a :, then a (possibly null) list of prerequisite files or dependencies. Text following a ; and all following lines that begin with a tab are shell commands to be executed to update the target. The first non-empty line that does not begin with a tab or # begins a new dependency or macro definition. Shell commands may be continued across lines with the <backslash><new-line> sequence. Everything printed by **make** (except the initial tab) is passed directly to the shell as is. Thus,

```
echo a\
b
```

will produce

```
ab
```

exactly the same as the shell would.

Sharp (#) and new-line surround comments.

The following *makefile* says that **pgm** depends on two files *a.o* and *b.o*, and that they in turn depend on their corresponding source files (*a.c* and *b.c*) and a common file *incl.h*:

```
pgm: a.o b.o
        cc a.o b.o -o pgm
a.o: incl.h a.c
        cc -c a.c
b.o: incl.h b.c
        cc -c b.c
```

Command lines are executed one at a time, each by its own shell. The **SHELL** environment variable can be used to specify which shell **make** should use to execute commands. The default is **/bin/sh**. The first one or two characters in a command can be the following: –, **@**, **–@**, or **@–**. If **@** is present, printing of the command is suppressed. If – is present, **make** ignores an error. A line is printed when it is executed unless the **–s** option is present, or the entry **.SILENT:** is in *makefile*, or unless the initial character sequence contains a **@**. The **–n** option specifies printing without execution; however, if the command line has the string **$(MAKE)** in it, the line is always executed (see discussion of the **MAKEFLAGS** macro under **Environment**). The **–t** (touch) option updates the modified date of a file without executing any commands.

Commands returning non-zero status normally terminate **make**. If the **–i** option is present, or the entry **.IGNORE:** appears in *makefile*, or the initial

character sequence of the command contains –, the error is ignored. If the **–k** option is present, work is abandoned on the current entry, but continues on other branches that do not depend on that entry.

The **–b** option allows old makefiles (those written for the old version of *make*) to run without errors.

Interrupt and quit cause the target to be deleted unless the target is a dependent of the special name **.PRECIOUS.**

Environment

The environment is read by **make**. All variables are assumed to be macro definitions and processed as such. The environment variables are processed before any *makefile* and after the internal rules. Thus, macro assignments in a makefile override environment variables. The **–e** option causes the environment to override the macro assignments in a makefile. Suffixes and their associated rules in the *makefile* override any identical suffixes in the built-in rules.

The **MAKEFLAGS** environment variable is processed by **make** as containing any legal input option (except **–f** and **–p**) defined for the command line. Further, upon invocation, **make** invents the variable if it is not in the environment, puts the current options into it, and passes it on to invocations of commands. Thus, **MAKEFLAGS** always contains the current input options. This proves very useful for super-makes. In fact, as noted above, when the **–n** option is used, the command **$(MAKE)** is executed anyway; hence, one can perform a **make –n** recursively on a whole software system to see what would have been executed. This is because the **–n** is put in **MAKEFLAGS** and passed to further invocations of **$(MAKE)**. This is one way of debugging all of the makefiles for a software project without actually doing anything.

Include Files

If the string *include* appears as the first seven letters of a line in a *makefile* and is followed by a blank or a tab, the rest of the line is assumed to be a filename and will be read by the current invocation, after substituting for any macros.

Macros

Entries of the form *string1* = *string2* are macro definitions. *String2* is defined as all characters up to a comment character or an unescaped new-line. Subsequent appearances of $(*string1*[:*subst1*=[*subst2*]]) are replaced by *string2*. The parentheses are optional if a single character macro name is used and there is no substitute sequence. The optional :*subst1*=*subst2* is a substitute sequence. If it is specified, all non-overlapping occurrences of *subst1* in the named macro are replaced by *subst2*. Strings (for the purposes of this type of substitution) are delimited by blanks, tabs, new-line characters, and beginnings of lines. An example of the use of the substitute sequence is shown under **Libraries.**

Internal Macros

There are five internally maintained macros that are useful for writing rules for building targets.

$*** The macro **$* stands for the filename part of the current dependent with the suffix deleted. It is evaluated only for inference rules.

$@** The **$@ macro stands for the full target name of the current target. It is evaluated only for explicitly named dependencies.

$<** The **$< macro is only evaluated for inference rules or the **.DEFAULT** rule. It is the module that is out-of-date with respect to the target (i.e., the manufactured dependent filename). Thus, in the **.c.o** rule, the **$<** macro would evaluate to the *.c* file.

An example for making optimized *.o* files from *.c* files is as follows:

```
.c.o:
        cc -c -O $*.c
```

or:

```
.c.o:
        cc -c -O $<
```

$?** The **$? macro is evaluated when explicit rules from the *makefile* are evaluated. It is the list of prerequisites that are out-of-date with respect to the target; essentially, those modules which must be rebuilt.

$% The **$%** macro is only evaluated when the target is an archive library member of the form *lib(file.o)*. In this case, **$@** evaluates to *lib* and **$%** evaluates to the library member, *file.o*.

Four of the five macros can have alternative forms. When an uppercase **D** or **F** is appended to any of the four macros, the meaning is changed to "directory part" for **D** and "file part" for **F**. Thus, **$(@D)** refers to the directory part of the string **$@**. If there is no directory part, *./* is generated. The only macro excluded from this alternative form is **$?**.

Suffixes

Certain names (for instance, those ending with *.o*) have inferable prerequisites such as *.c*, *.s*, etc. If no update commands for such a file appear in *makefile*, and if an inferable prerequisite exists, that prerequisite is compiled to make the target. In this case, **make** has inference rules which allow building files from other files by examining the suffixes and determining an appropriate inference rule to use. The current default inference rules are as follows:

```
.c .c~ .f .f~ .sh .sh~
.c.o .c.a .c~.o .c~.c .c~.a
.f.o .f.a .f~.o .f~.f .f~.a
.h~.h .s.o .s~.o .s~.s .s~.a .sh~.sh
.l.o .l.c .l~.o .l~.l .l~.c
.y.o .y.c .y~.o .y~.y .y~.c
```

The internal rules for **make** are contained in the source file *rules.c* for the **make** program. These rules can be locally modified. To print out the rules compiled into the **make** on any machine in a form suitable for recompilation, the following command is used:

```
make -fp - 2>/dev/null </dev/null
```

A tilde in the above rules refers to an SCCS file [see **sccsfile**(4)]. Thus, the rule **.c~.o** would transform an SCCS C source file into an object file (*.o*). Because the **s.** of the SCCS files is a prefix, it is incompatible with **make**'s suffix point of view. Hence, the tilde is a way of changing any file reference into an SCCS file reference.

A rule with only one suffix (i.e., **.c:**) is the definition of how to build *x* from *x.c*. In effect, the other suffix is null. This is useful for building targets from only one source file (e.g., shell procedures, simple C programs).

Additional suffixes are given as the dependency list for SUFFIXES. Order is significant; the first possible name for which both a file and a rule exist is inferred as a prerequisite. The default list is:

.SUFFIXES: .o .c .c~ .y .y~ .l .l~ .s .s~ .sh .sh~ .h .h~ .f .f~

Here again, the above command for printing the internal rules will display the list of suffixes implemented on the current machine. Multiple suffix lists accumulate; **.SUFFIXES:** with no dependencies clears the list of suffixes.

Inference Rules

The first example can be done more briefly.

```
pgm: a.o b.o
	cc a.o b.o -o pgm
a.o b.o: incl.h
```

This is because **make** has a set of internal rules for building files. The user may add rules to this list by simply putting them in the *makefile*.

Certain macros are used by the default inference rules to permit the inclusion of optional matter in any resulting commands. For example, **CFLAGS**, **LFLAGS**, and **YFLAGS** are used for compiler options to **cc**(1), **lex**(1), and **yacc**(1), respectively. Again, the previous method for examining the current rules is recommended.

The inference of prerequisites can be controlled. The rule to create a file with suffix *.o* from a file with suffix *.c* is specified as an entry with **.c.o:** as the target and no dependents. Shell commands associated with the target define the rule for making a *.o* file from a *.c* file. Any target that has no slashes in it and starts with a dot is identified as a rule and not a true target.

Libraries

If a target or dependency name contains parentheses, it is assumed to be an archive library, the string within parentheses referring to a member within the library. Thus *lib(file.o)* and *$(LIB)(file.o)* both refer to an archive library that contains *file.o*. (This assumes the **LIB** macro has been previously defined.) The expression *$(LIB)(file1.o file2.o)* is not legal. Rules pertaining to archive libraries have the form *.XX***.a** where the *XX* is the suffix from which the archive member is to be made. An unfortunate byproduct of the current implementation requires the *XX* to be different from the suffix of the archive member. Thus, one cannot have *lib(file.o)* depend upon *file.o* explicitly. The most common use of the archive interface follows. Here, we assume the source files are all C type source:

```
lib:  lib(file1.o) lib(file2.o) lib(file3.o)
	@echo lib is now up-to-date
.c.a:
	$(CC) -c $(CFLAGS) $<
	$(AR) $(ARFLAGS) $@ $*.o
	rm -f $*.o
```

In fact, the **.c.a** rule listed above is built into **make** and is unnecessary in this example.

A more interesting, but more limited example of an archive library maintenance construction follows:

```
lib:  lib(file1.o) lib(file2.o) lib(file3.o)
      $(CC) -c $(CFLAGS) $(?:.o=.c)
      $(AR) $(ARFLAGS) lib $?
      rm $?   @echo lib is now up-to-date
.c.a:;
```

Here the substitution mode of the macro expansions is used. The **$?** list is defined to be the set of object filenames (inside *lib*) whose C source files are out-of-date. The substitution mode translates the *.o* to *.c*. (Unfortunately, one cannot as yet transform to **.c~**; however, this may become possible in the future.) Note also, the disabling of the **.c.a:** rule, which would have created each object file, one by one. This particular construct speeds up archive library maintenance considerably. This type of construct becomes very cumbersome if the archive library contains a mix of assembly programs and C programs.

Parallel Support

make includes a parallel processing ability. If the string separating a target from its prerequisites is **:&, :$&, ::&, or ::$&, make** can run the command sequences to make the prerequisites simultaneously. If two names are separated by an ampersand (**&** or **$&**) on the right side of a colon, those two may be created in parallel.

In this example *makefile*, if the objects *a.o*, *b.o*, and *c.o* are out of date, they are built in parallel:

```
pgm: $&      a.o b.o c.o
      cc -o pgm a.o b.o c.o
a.o:  a.c defs.h
b.o:  b.c defs.h
c.o:  c.c defs.h
```

In this example, only *a.o* and *b.o* are built in parallel, with *c.o* being started after the other two complete:

```
pgm:  a.o $& b.o c.o
      cc -o pgm a.o b.o c.o
a.o:  a.c defs.h
b.o:  b.c defs.h
c.o:  c.c defs.h
```

Since the **-P** option is not compatible with other versions of **make**, the use of **$&** is preferred over **&**. This is because **$&** is a macro which is replaced by **&**, but in other versions of **make** it has no effect. Thus, the same *makefile* can be used on other systems. Another way to avoid problems is by creating a shell script called **make**, and placing it in */usr/local*:

```
#!/bin/sh
```

```
if [ $# != 0 ]
then
        exec /bin/make P='&' "$@"
else
        exec /bin/make P='&'
fi
```

and changing your targets from their present construct:

```
FILES : headers.o main.o
```

to a portable form:

```
FILES :$(P) headers.o main.o
```

If you use either technique, **make** is portable between systems. The **-P** option controls the number of simultaneous command streams. **make** invoked with an argument of **-P1** selects no parallelism, and is useful when error messages from parallel command sequences are intermingled, obscuring the actual source of error. (A useful technique for searching the output for error messages is to look for lines that do not begin with a tab character.)

To execute command sequences in parallel, **make** forks off processes without waiting for them, and indicates they've been forked by appending an ampersand and the process ID while printing the command (similar to using the ampersand to indicate asynchronous jobs in the shell). If the forked-off command sequence returns a non-zero exit status, **make** precedes the error (or warning) message with the process ID, to help determine the source of the error.

Not all command sequences can be made parallel. Some command sequences generate filenames in the current directory that do not depend on the file being built. For example, operations involving two or more usages of **yacc**, **lex**, **ar**, or **xstr** in the same directory cannot be made parallel.

While building the prerequisites in parallel, only the last (or only) command in a multiple command sequence is eligible to be forked for asynchronous execution. Other commands are executed sequentially. For example, in the following *makefile*, only *lastcommand* is run in parallel; *firstcommand* and *secondcommand* are processed before any of the commands to build *b* are invoked:

```
pgm:$& a b
a:
        firstcommand
        secondcommand
        lastcommand
b:
        b-firstcommand
        b-secondcommand
        b-lastcommand
```

To circumvent this, make all the commands be considered as one long shell

command, as in:

```
pgm:$& a b
a:
        firstcommand; \
        secondcommand; \
        lastcommand
b:
        b-firstcommand; \
        b-secondcommand; \
        b-lastcommand
```

The backslash at the end of a command line causes the following line to be regarded as a continuation. A *makefile* set up in this way has all the commands to build *a* be run in parallel with all the commands to build *b*. Note that all three commands to build *a* are now executed by the same shell (rather than three separate invocations of the shell). Commands interpreted by the shell directly (to change directory, for example) are no longer isolated to one particular command line. This should be taken into consideration when altering the makefiles in this way.

FILES

[Mm]akefile and s.[Mm]akefile
/bin/sh

SEE ALSO

cc(1), cd(1), lex(1), sh(1), yacc(1), printf(3S), sccsfile(4)

NOTES

Some commands return non-zero status inappropriately; use **-i** to overcome the difficulty.

BUGS

Filenames with the characters = : @ will not work. Commands that are directly executed by the shell, notably **cd**(1), are ineffectual across newlines in **make**. The syntax *(lib(file1.o file2.o file3.o)* is illegal. You cannot build *lib(file.o)* from *file.o*. The macro **$(a:.o=.c~)** does not work. Named pipes are not handled well.

NAME

makealiases, makewhatis – generate man page databases

SYNOPSIS

/usr/lib/makealiases

/usr/lib/makewhatis

DESCRIPTION

makealiases scans the files in */usr/catman, and creates the alias file /usr/catman/aliases.* **makewhatis** also scans the files in */usr/catman* and creates the file */usr/catman/whatis* for use by */usr/bin/whatis, /usr/bin/apropos,* and */usr/bin/man.* Any time new manual pages are installed within */usr/catman,* the aliases and whatis databases should be regenerated.

FILES

/usr/catman/aliases	the man page aliases
/usr/catman/whatis	the whatis database
/usr/catman/man?/*	preformatted manual pages

SEE ALSO

man(1), whatis(1), apropos(1)

NAME

makedev – make system special files

SYNOPSIS

/dev/MAKEDEV [–n] *device...*

DESCRIPTION

MAKEDEV is a shell script used to install special files. It resides in the */dev* directory, the usual location of special files. All device special files are created relative to the current directory. You must change directories to the target location before running **MAKEDEV.**

If the **–n** option is specified as the first argument to **MAKEDEV**, the actions are only printed and not executed. Since all devices are created using **mknod**(1M), **MAKEDEV** (without the **–n** option) can be used only by the superuser.

Arguments to **MAKEDEV** are usually of the form *device-name** where *device-name* is one of the supported devices listed in section 7 of the manual and the asterisk (*) represents a logical unit number. In the following example, **zd** is the *device name* and **0** is the logical unit number.

```
# cd /dev
# ./MAKEDEV zd0
```

The following special arguments create assorted collections of devices:

auto Create device nodes for all devices found by the autoconfigure process at boot time. The devices created are limited only by the kernel I/O configuration.

std Create the standard devices for the system. These devices are */dev/console, /dev/syscon, /dev/altcon, /dev/consmem, /dev/tty, /dev/mem, /dev/kmem, /dev/null, /dev/swap, /dev/usclk, /dev/log, /dev/getpty, /dev/spx, /dev/cmpt,* and */dev/vtoc.*

local Create those devices specific to the local site. This request causes the shell script */dev/MAKEDEV.local* to be executed. Any site specific commands should be included in this file.

DIAGNOSTICS

Messages are either self-explanatory, or are generated by one of the programs called from the script.

SEE ALSO

sh (1), config(1M), mknod(1M), intro(7)

NAME

makekey – generate encryption key

SYNOPSIS

/usr/lib/makekey

DESCRIPTION

makekey improves the usefulness of encryption schemes that depend on a key by increasing the amount of time required to search the key space. It first attempts to read 8 bytes for its *key* (the first 8 input bytes), then it attempts to read 2 bytes for its *salt* (the last 2 input bytes). The output depends on the input in a way intended to be difficult to compute (that is, to require a substantial fraction of a second).

The first 8 input bytes (the input key) can be arbitrary ASCII characters. The last 2 (the *salt*) are best chosen from the set of digits, ., /, and upper- and lowercase letters. The salt characters are repeated as the first two characters of the output. The remaining 11 output characters are chosen from the same set as the salt and constitute the output key.

The transformation performed is essentially the following: the salt is used to select one of 4,096 cryptographic machines all based on the National Bureau of Standards DES algorithm, but broken in 4,096 different ways. Using the input key as key, a constant string is fed into the machine and recirculated a number of times. The 64 bits that come out are distributed into the 66 output key bits in the result.

makekey is intended for programs that perform encryption. Usually, its input and output will be pipes.

SEE ALSO

ed(1), crypt(1), vi(1), passwd(4)

NOTES

makekey can produce different results depending upon whether the input is typed at the terminal or redirected from a file.

This command is available only in the United States.

NAME

man – display on-line man pages

SYNOPSIS

man **–k** keyword ...
man **–f** file ...
man [**–P** path] [–] [section] title ...

DESCRIPTION

The **man** program provides on-line access to the man pages in the *Reference Manual*. When the **man** command is issued, it formats a specified set of manual pages.

When given the option **–k** and a set of keywords, **man** prints out a one-line synopsis of each manual section whose listing in the table of contents contains those keywords. This is the same as **apropos**(1).

When given the option **–f** and a list of *file*s, **man** attemps to locate manual sections related to those files, printing out the table of contents lines for those sections. This is the same as **whatis**(1).

When neither **–k** nor **–f** is specified, **man** formats a specified set of manual pages. If a section identifier is given, **man** looks in that section of the manual for the given *title*s.

The **–P** option allows you to specify a *path* for the location of the man pages to be displayed.

section is an Arabic section number such as 1. The number can be followed by a one-letter classifier (such as *1g*, which indicates a graphics program in section 1) or by a three-letter classifier (such as *3pps*, which indicates a routine in the Parallel Programming Library in section 3).

If *section* is omitted, **man** searches all sections of the manual, giving preference to commands over subroutines in system libraries, and printing the first section it finds, if any.

If the standard output is a terminal or if the flag – is given, **man** pipes its output through **cat**(1) and through **pg**(1) to cause output to stop after each page. Press Return to continue scrolling the output.

The use of **pg**(1) as the pager can be changed by setting the environment variable **PAGER**.

Manual pages aliases are kept in the */usr/catman/aliases* file.

FILES

/usr/catman/man?/*
/usr/catman/aliases

SEE ALSO

pg(1), apropos(1), whatis(1)

BUGS

The manual is supposed to be reproducible either on the phototypesetter or on a typewriter. However, on a typewriter some information is necessarily lost.

NAME

mcs – manipulate the object file comment section

SYNOPSIS

mcs [options] object-file ...

DESCRIPTION

The **mcs** command manipulates the comment section (normally *.comment*) in an object file. It is used to add to, delete, print, and compress the contents of the comment section in an operating system object file. **mcs** must be given one or more of the options described below. It takes each of the options given and applies them in order to the *object-files*.

If the object file is an archive, the file is treated as a set of individual object files. For example, if the **–a** option is specified, the string is appended to the comment section of each archive element.

The following options are available.

–a *string* Append *string* to the comment section of the *object-files*. If *string* contains embedded blanks, it must be enclosed in quotation marks.

–c Compress the contents of the comment section. All duplicate entries are removed. The ordering of the remaining entries is not disturbed.

–d Delete the contents of the comment section from the object file. The object file comment section header is removed also.

–n *name* Specify the name of the section to access. By default, **mcs** deals with the section named *.comment*. This option can be used to specify another section.

–p Print the contents of the comment section on the standard output. If more than one name is specified, each entry printed is tagged by the name of the file from which it was extracted, using the format *filename:string*.

EXAMPLES

mcs –p *file* # Print *file*'s *comment* section

mcs –a *string file* # Append *string* to *file*'s *comment* section

FILES

TMPDIR/mcs* Temporary files

TMPDIR/* Temporary files

TMPDIR is usually */usr/tmp*, but can be redefined by setting the environment variable **TMPDIR** [see **tempnam**() in **tmpnam**(3S)].

SEE ALSO

cpp(1), a.out(4)

NOTES

mcs cannot add new sections or delete existing sections to executable objects with magic number 0413 [see **a.out**(4)].

NAME

menu – menu processor

SYNOPSIS

menu [menuname]

DESCRIPTION

The **menu** command initiates the ptx/ADMIN menu system. The initial menu can be called from the command line or it can be defined by the **TFBEGMENU** environment variable.

menuname Name of the initial menu.

The initial menu must be defined as an entry point in the global security table. The System Administrator uses this command to control where each user enters the menu system.

The menu system looks for three optional environment control files before displaying the user's initial menu. These three files are used to establish fixed values for environment variables unique to the menu system. The environment variables appear in the following format:

name=value

The first of these files, */etc/tfenv*, controls system-wide values that do not vary with each user and identifies where the components of the menu system are stored.

For example, */etc/tfenv* might include the following entries:

```
TFADMBIN=/usr/tf/admbin
TFADMFILE=/usr/tf
PATH=$PATH:$TFBIN:$TFADMBIN
```

Note that the **PATH** variable is used by the menu system and by the shell to locate programs.

The *$TFADMFILE/etc/tfenv* file establishes variables that control (that is, activate and deactivate) features in the menu system. This is an optional file. The environment variables that are normally in this file can be placed in the */etc/tfenv* file, if desired. The *$TFADMFILE/etc/tfenv* contains the following environment variables:

```
TFBEGMENU=clmain
TFDSPCODE=No
TFDSPMENU=Multiple
```

The last file that the menu system reads is the user's file *$HOME/.tfenv*. This file contains environment variables that are tailored for the user. For example, the file might contain the following entries:

```
TFBEGMENU=mymenu
TFDSPCODE=Yes
```

These three control files (*/etc/tfenv*, *$TFADMFILE/etc/tfenv*, and *$HOME/.tfenv*) allow the administrator to establish a user account with the

menu shell, **tfmenush**, as the user's initial program. It does not require any additional setup or modification in the user's home directory.

Environment Variables

TFTERM Identifies the terminal type.

TERM If **TFTERM** is not set, this variable identifies the terminal type.

TERMINFO Identifies the location of the terminfo terminal descriptions. Normally you do not need to define this environment variable since the directory *$TFADMFILE/terminfo* is used by default.

TFBEGMENU If the command line does not include the name of an initial menu, this variable identifies the initial menu for the user.

TFBIN Identifies the directory where the menu system user programs are located.

TFADMBIN Contains the name of the directory where menu system binaries are stored. The System Administrator is the only authorized user. In some installations, **TFBIN** and **TFADMBIN** are set to the same value.

TFADMFILE Contains the name of the head directory where the menu system subdirectories are stored. There are four important directories:

etc	control information
misc	error logs and dead messages
edc	compiled forms (in most cases)
menu	menu definitions (in most cases)

TFEDCPATH Identifies directories that are alternate locations for the command description files. Directory names are separated with a colon in the string, similar to the **PATH** environment variable.

TFMENUPATH Identifies a list of directories that are alternate locations for the menu definition files. Directory names are separated with a colon in the string similiar to the **PATH** environment variable.

TFDSPMENU Identifies if the menu driver displays the top menu in the menu stack (**TFDSPMENU**=Single), or uses the graphics mode to tile the top three menus on the menu stack (**TFDSPMENU**=Multiple). If this environment variable is not defined, the menu system defaults to the graphics mode.

FILES

${TERMINFO}/[a-z]/termtype Terminal definition file

${TFADMFILE}/menu/*menuname*	Default menu definition file
/etc/passwd	System password file
/etc/group	System group file
/etc/tfenv	Establishes menu system home directories and other system-wide parameters
$TFADMFILE/etc/tfenv	Set menu system feature variables
$HOME/.tfenv	Override menu system feature variables
${TFADMFILE}/etc/task	Global security table
${TFADMFILE}/menumsg	System message files
${TFADMFILE}/misc/dead.msg	System dead message file
${TFADMFILE}/misc/errorlog	System error message log

SEE ALSO

prompt(1M)

NAME

menumsg – route user message to menu message queue

SYNOPSIS

menumsg [**–u** *user*] [**–t** *term*] [*message*]

DESCRIPTION

The **menumsg** command allows a program or user to send a message to another user via the menu message queue.

If the recipient is not logged onto the menu system, the messages are held in the queue until the recipient executes the menu system.

–u *user* Identifies the user who will receive the message. The user name must exist in the system password file (*/etc/passwd*). If the command does not include a user name, the message is sent to the user who executed the **menumsg** command. Selection of a user name is optional.

–t *term* Identifies the terminal that will receive the message. If the command does not include a terminal id, the message is sent to the terminal that executed the **menumsg** command. Selection of a terminal is optional.

message Text of the message. If the command does not include message text, the standard input is read. Message text is required.

FILES

/etc/passwd	System password file
${TFADMFILE}/menumsg	System message files
${TFADMFILE}/misc/dead.msg	System dead message file

SEE ALSO

menu(1M), tfnotify(1M)

NAME

merge – three-way file merge

SYNOPSIS

merge [**-p**] file1 file2 file3

DESCRIPTION

merge incorporates all changes that lead from *file2* to *file3* into *file1*. The result goes to standard output if **-p** is present, into *file1* otherwise. **merge** is useful for combining separate changes to an original. Suppose *file2* is the original, and both *file1* and *file3* are modifications of *file2*. Then **merge** combines both changes.

An overlap occurs if both *file1* and *file3* have changes in a common segment of lines. **merge** prints how many overlaps occurred and includes both alternatives in the result. The alternatives are delimited as follows:

```
<<<<<<< file1
lines in file1
=======
lines in file3
>>>>>>> file3
```

If there are overlaps, the user should edit the result and delete one of the alternatives.

SEE ALSO

co(1), diff3(1), diff(1), rcsmerge(1)

NOTES

Copyright 1982 by Walter F. Tichy.

NAME

mesg – permit or deny messages

SYNOPSIS

mesg [**–n**] [**–y**]

DESCRIPTION

mesg with argument **n** forbids messages via **write**(1) by revoking non-user write permission on the user's terminal. **mesg** with argument **y** reinstates permission. All by itself, **mesg** reports the current state without changing it.

FILES

/dev/tty*

SEE ALSO

write(1)

DIAGNOSTICS

Exit status is 0 if messages are receivable, 1 if messages are not receivable, 2 on error.

NAME

message – puts its arguments on message line

SYNOPSIS

message [**–t**] [**–p**] [**–o**] [**–b** [n]] "[string]"

DESCRIPTION

The **message** command outputs its string arguments to the message line. If there is no string, the *stdin* input to **message** is used. Specifying **–t** causes the message to be removed after the next keypress. This is the default. Specifying **–p** causes the message to be displayed until another message is displayed. The **–o** flag forces **message** to "tee" its message to *stdout*. The **–b** [*n*], where *n* is an integer from 1 to 10, rings the terminal bell *n* times. The default is 1. If the terminal has no bell, the screen is flashed instead, if possible.

If the **message** command is being used solely for the bell or working indicator control, remember to give it a null string argument unless input is being piped to it. The string should always be the last argument.

EXAMPLES

When the value entered in the field is wrong, ring the bell 3 times and then put up the invalid field message "Try again!"

```
invalidmsg='message -b 3 "" 'Try again!
```

Put out a message to tell the user what is being done:

```
done='message hello has been set in your environment'
```

NAME

mkdir – make directories

SYNOPSIS

mkdir [**–m** mode] [**–p**] dirname ...

DESCRIPTION

mkdir creates the named directories in mode 777. The mode will be altered by **umask**(1) unless the **–m** option is specified.

Standard entries in a directory (the files . for the directory itself and .. for its parent) are made automatically. **mkdir** cannot create these entries by name. Creation of a directory requires write permission in the parent directory.

The owner ID and group ID of the new directories are set to the process's real user ID and the group ID of the parent directories, respectively.

Two options apply to **mkdir**:

–m This option allows users to specify the mode to be used for new directories. Choices for modes can be found in **chmod**(1).

–p With this option, **mkdir** creates *dirname* by creating all the non-existing parent directories first.

EXAMPLES

To create the subdirectory structure *ltr/jd/jan*, type this command:

mkdir –p ltr/jd/jan

SEE ALSO

csh(1), ksh(1), sh(1), rm(1), umask(1), intro(2), mkdir(2)

DIAGNOSTICS

mkdir returns exit code 0 if all directories given in the command line were made successfully. Otherwise, it prints a diagnostic and returns non-zero. An error code is stored in *errno*.

NAME

mkfs – construct a filesystem

SYNOPSIS

/etc/mkfs special size [nsect [ntrack [blksize [fragsize [ncpg [minfree [rps [nbpi]]]]]]]]

DESCRIPTION

mkfs constructs a filesystem by writing on the special file *special*. **mkfs** provides an alternate interface to the **newfs**(1M) command, and can be used to initialize file systems on disks which have no VTOC. The numeric size specifies the number of sectors in the filesystem. **mkfs** builds a filesystem with a root directory and a *lost+found* directory (see **fsck**(1M)). The number of inodes is calculated as a function of the filesystem size.

The following optional arguments allow finer control over the parameters of the filesystem:

nsect Specifies the number of sectors per track on the disk.

ntrack Specifies the number of tracks per cylinder on the disk.

blksize Gives the primary block size for files on the filesystem. It must be a power of two, selected from 4096 or 8192.

fragsize Gives the fragment size for files on the filesystem. Represents the smallest amount of disk space that will be allocated to a file. It must be a power of two, selected from the range 512 to 8192.

ncpg Specifies the number of disk cylinders per cylinder group.

minfree Specifies the minimum percentage of free disk space allowed. Once the filesystem capacity reaches this threshold, only the superuser is allowed to allocate disk blocks. The default value is 10%.

rps Specifies the number of revolutions per second, if a disk does not revolve at 60 revolutions per second.

nbpi Specifies the number of bytes per inode, if a filesystem will have more or less than the average number of files. Used to increase or decrease the number of inodes that are created.

SEE ALSO

newfs(1M), fsck(1M), tunefs(1M), dir(4), fs(4)

BUGS

There should be some way to specify bad blocks.

NAME

mknod – build special file

SYNOPSIS

/etc/mknod name **b** | **c** major minor
/etc/mknod name **p**

DESCRIPTION

mknod makes a directory entry and corresponding inode for a special file.

The first argument is the *name* of the entry. The operating system convention is to keep such files in the */dev* directory.

In the first case, the second argument is **b** if the special file is block-type (disks, tape) or **c** if it is character-type (other devices). The last two arguments are numbers specifying the *major* device type and the *minor* device (e.g., unit, drive, or line number). They may be either decimal or octal. The assignment of major device numbers is specific to each system. The information is contained in the system source file *conf.c*. You must be the superuser to use this form of the command.

The second case is the form of the **mknod** that is used to create FIFO's (a.k.a named pipes).

SEE ALSO

mknod(2)

NAME

mkpwdbm – make a database for fast password lookup

SYNOPSIS

mkpwdbm [-fvps]

DESCRIPTION

The **mkpwdbm** program generates a database that allows fast indexing into the */etc/passwd* file. If the optional */etc/shadow* file exists, **mkpwdbm** also generates a database allowing fast lookups into this file.

If your machine has a large number of names (> 500) in the */etc/passwd* file, using the database files will speed lookups making use of the library routines that search these files.

This command can be run only by the superuser. Normally it is run by a daemon program (**/etc/pwdhd**) and is not invoked manually. The options are as follows:

–f Force creation of the database files. Normally the files will be updated only if they presently exist.

–v Verbose option. The command is intended to be run by a daemon and typically generates no output. Using either the **–f** or the **–v** command line options allows error messages to be generated. The **–v** option also prints a count of names found in the data files. If the command finds duplicate names or user ids, they are displayed.

–p Rebuild only the database for */etc/passwd*. By default, the databases for both */etc/passwd* and */etc/shadow* are built.

–s Rebuild only the database for */etc/shadow*. By default, the databases for both */etc/passwd* and */etc/shadow* are built.

FILES

/etc/passwd
/etc/passwd_dbm.dir
/etc/passwd_dbm.pag
/etc/shadow
/etc/shadow_dbm.dir
/etc/shadow_dbm.pag

SEE ALSO

/etc/pwdhd

BUGS

If you edit the password or shadow files by hand to add a user and do not alter the file byte count, the library routines might not be able to find the new user. Because other problems not related to the hashed database can occur when the files are edited by hand, you should avoid editing the files in this manner.

NAME

mkvtoc – populate VTOC on disks

SYNOPSIS

/etc/mkvtoc [**–d** arguments] [**–g**] [**–f**] [**–s** datafile] disk disktype

DESCRIPTION

mkvtoc populates the VTOC (volume table of contents) on disks.

The following options apply to **mkvtoc:**

–d Permits change to a single partition. *arguments* consist of a colon-separated list of entries similar to a line from *datafile* (described below).

–g The VTOC is initialized and the disk geometry is set from the file **/etc/diskinfo/***disktype***.geom**. This option must be supplied if the disk does not have a VTOC present.

–f Allow partitions of type **V_RESERVED** to be changed. Use of this option is strongly discouraged.

–s The VTOC is populated according to a *datafile* created by the user. The *datafile* format is described below.

If no options are given, the VTOC is populated according to the specification of a default file, *disktype* in directory */etc/vtoc,* where *disktype* was supplied on the command line. For example, for a 792-megabyte disk, the *disktype* specific would be **m2382k** and the default file used is */etc/vtoc/m2382k* .

The *disk* on which the VTOC is installed may be specified as a simple disk name such as **zd0** or **sd1** without any path prefix. If this form of *disk* is used, the appropriate path to a special device is generated automatically. Alternatively, the pathname to a character special device should be given. The character special device may be a disk partition, the disk's diagnostic device, or the prepartitioned disk device. (See **vtoc**(7)).

The *datafile* contains one specification line for each partition, starting with partition 0. Lines beginning with an asterisk are treated as comments. A specification line is composed of position-dependent entries separated by white space. A line has the following format:

partition type starting_sector size_in_sectors block_size frag_size

The entries have the following values:

partition The partition number: 0-254.

type The partition type: a decimal number. The following are valid types: V_NOPART: 0, V_RAW: 1, V_BOOT: 2, V_RESERVED: 3, V_FW: 4.

starting sector The sector number on which the partition starts.

size in sectors The number of sectors occupied by the partition.

The partition types have the following meanings:

V_NOPART This partition number does not represent valid, useable disk space. A partition may be marked as **V_NOPART** to serve as a placeholder. Note that if the datafile contains less than the maximum possible number of partitions, the remainder are marked **V_NOPART** .

V_RAW This is a regular partition.

V_BOOT This is an area of disk reserved for a bootstrap. As such, filesystems should not be mounted here. **V_BOOT** is a descriptive tag and has no functional difference from **V_RAW** .

V_RESERVED This is an area of the disk reserved for some disktype-specific purpose, such as bad block lists, diagnostic tracks, or the VTOC itself. Partitions of type **V_RESERVED** are treated differently than partitions of other types in that they may not be changed via the **mkvtoc** command, unless the **-f** flag is specified on the command line. In general it is not wise to alter **V_RESERVED** partitions in any way, since various parts of the system depend on these disk sectors to be unchanged.

V_FW This is an area of disk reserved for system controller firmware. As such, filesystems should not be mounted here. **V_FW** is a descriptive tag and has no functional difference from **V_RAW** .

A number of sanity checks are performed on the new VTOC before it is written to disk. For example, the block size and fragment sizes are checked against system norms and the total number of sectors in partitions are checked to make sure they do not exceed the size of the disk. In addition, if any two partitions have sectors which overlap, the **mkvtoc** command will fail.

SEE ALSO

prtvtoc(1M)

WARNINGS

If a disk which has not previously contained a VTOC has data present in sectors 16 through 31, that data will be overwritten by the VTOC.

NAME

monitor – monitor system activity

SYNOPSIS

/etc/monitor [**–c** *n*] [**–i** *n*] [**–p** *n*] [**–t**] [**–f**]

DESCRIPTION

monitor displays various measures of system activity. The first portion of the display is a set of bar graphs of processor activity, one for each processor plus a graph for the system labeled "Total." Each bar indicates the percentage of time the respective processor was busy and is composed of two symbols, '=' for system time and '-' for user time. The rest of the display contains a number of user selectable groups of system activity monitors. To exit **monitor**, a Return, Linefeed, or the letter **q** may be typed. A Ctrl-L will cause the screen to be redrawn.

monitor's command-line options are as follows:

–c *n* Specifies the number of columns to use for displaying the processor bar graphs. Valid choices for *n* are 1, 2, and 3.

–i *n* Specifies the length of the interval (in seconds) to sleep between updates of the screen. The default is 1.

–p *n* If the **monitor** process is running with sufficient privileges, it will "affinity" itself to processor *n*. The default is "no affinity", i.e. the operating system schedules **monitor** like any other process, running it across all processors. (See **tmp_affinity**(2SEQ)).

–t Specifies that terse activity monitor descriptions should be used. This allows more information to fit on the screen.

–f Indicates that two screens are to be used; one for the processor graphs and one for the activity monitors. While **monitor** is running, typing an **f** will flip between the two screens.

Activity monitors are as follows (labels in **–t** format are given in parentheses):

Total User Time (usr %)
Time the system spent in user mode expressed as percent of one processor.

Total System Time (sys %)
Time the system spent in system mode expressed as percent of one processor.

Total Time (tot %)
Sum of user and system time expressed as percent of one processor.

Number of Procs (procs)
Number of processes in the system (occupied process table entries).

Running Procs (on p)
Number of processes currently running on a processor.

Runnable Procs (runq)
Number of processes that are runnable and waiting on the run queue to be dispatched to a processor.

Fast Wait (wait)
Number of processes that are waiting on disk I/O, page I/O, or a kernel resource.

Sleeping Procs (sleep)
Number of processes that are sleeping.

Swapped Procs (swapped)
Number of processes that are swapped to disk.

Procs FS io Wait (fs io)
Number of processes waiting for filesystem I/O. These are included in FastWait total.

Procs Phy io Wait (phys io)
Number of processes waiting for raw disk I/O. These are included in FastWait total.

System Calls (sysc)
Number of system calls per second.

Context Switches (csw)
Number of context switches per second.

Interrupts (intr)
Number of interrupts per second.

Traps (traps)
Number of traps per second.

Forks (fork)
Number of forks per second.

Vforks (vfork)
Number of vforks per second.

Execs (exec)
Number of execs per second.

Disk Transfers (dk xf)
Number of disk transfers per second.

Disk KBytes (dk KB)
Number of kilobytes transferred per second.

TTY Chars In (ttyin)
Number of characters received on TTY lines per second.

TTY Chars Out (ttyout)
Number of characters transmitted on TTY lines per second.

Page Faults (pf)
Number of page faults per second.

Free Page Recs (pg rec)
Number of page reclaims from the free list per second. The sum of page reclaims from the free list and the dirty list equals the total reclaims (i.e. minor page faults).

Dirty Page Recs (pgdrec)
Number of page reclaims from the dirty list per second. The sum of page reclaims from the dirty list and the free list equals the total

reclaims (i.e. minor page faults).

Page Ins (pgin)
Number of page ins (major faults) per second.

Pages Paged In (ppgin)
Number of pages paged in per second.

Page Outs (pgout)
Number of page outs per second.

Pages Paged Out (ppgout)
Number of pages paged out per second.

Swap Ins (sw in)
Number of processes swapped in per second.

Pages Swapped In (pswin)
Number of pages swapped in per second.

Swap Outs (sw out)
Number of processes swapped out per second.

Pages Swapped Out (pswout)
Number of pages swapped out per second.

Deficit (deficit)
Virtual memory deficit in Kbytes

Free Memory (free)
Free memory in Kbytes.

Dirty Memory (dirty)
Dirty memory in Kbytes

Total Virtual Mem (tot vm)
Sum of virtual sizes of all processes.

FS Blk Reads (fs brd)
Number of actual disk reads initiated by the filesystem.

FS Blk Writes (fs bwt)
Number of actual disk writes initiated by the filesystem.

FS Read Hit (fs rdh)
Percentage of filesystem read requests that were satisfied from the buffer cache.

FS Write Hit (fs wth)
Percentage of filesystem write requests that did not result in an actual disk write.

Raw Reads (raw rd)
Number of raw device reads from all devices.

Raw Writes (raw wt)
Number of raw device writes from all devices.

Raw Read KB (raw rKB)
Number of kilobytes (KB) transferred via raw device reads.

Raw Write KB (raw wKB)
Number of kilobytes (KB) transferred via raw device writes.

Locks Used (lckcnt)
Number of record/file locks (see **lockf**(3C) or **fcntl**(2)) currently in use.

Percent Locks Used (%locks)
The percentage of available record/file locks (see **lockf**(3C) or **fcntl**(2)) currently in use.

Message Ops (msgops)
Number of System V message queue operations per second.

Semaphore Ops (semops)
Number of System V semaphore operations per second.

BUGS

monitor samples dynamically changing data. From time to time, some slight anomalies may appear. The most obvious one is that occasionally the number of processes on processor will be greater than the number of processors in the system.

NAME

more - view a file one full screen at a time

SYNOPSIS

more [**–cdflsurw**] [*–n*] [*+linenumber*] [*+/pattern*] [*name ...*]

DESCRIPTION

The **more** filter allows examination of continuous text one full screen at a time. It normally pauses after each full screen, printing "`--More--`" at the bottom of the screen. If the user then presses the RETURN key, one more line is displayed. If the user presses the SPACEBAR, another full screen is displayed. Other possibilities are described below.

The command line options include the following:

–n An integer which is the size (in lines) of the window which **more** is to use instead of the default.

–c **more** draws each page by beginning at the top of the screen and erasing each line just before it displays a new line. This avoids scrolling the screen, making it easier to read while **more** is writing. This option is ignored if the terminal does not have the ability to clear to the end of a line.

–d **more** prompts with the message "`Hit space to continue, Del to abort`" at the end of each full screen. This is useful if **more** is being used as a filter in some setting, such as a class, where many users may be inexperienced.

–f This option causes **more** to count logical lines instead of screen lines. Long lines are not folded. This option is recommended if **nroff** output is being piped through **ul**, since the latter may generate escape sequences. These escape sequences contain characters that would ordinarily occupy screen positions, but that do not print when they are sent to the terminal as part of an escape sequence. Thus, **more** may think that lines are longer than they actually are and fold lines erroneously.

–l Does not treat Ctrl-L (FORMFEED) specially. If this option is not given, **more** pauses after any line that contains a Ctrl-L, as if the end of a full screen had been reached. Also, if a file begins with a FORMFEED, the screen is cleared before the file is printed.

–s Squeezes multiple blank lines from the output, producing only one blank line. Especially helpful when viewing **nroff** output, this option maximizes the useful information present on the screen.

–u Normally, **more** handles underlining, such as that produced by **nroff** in a manner appropriate to the particular terminal. If the terminal can perform underlining or has a stand-out mode, **more** outputs appropriate escape sequences to enable underlining or stand-out mode for underlined information in the source file. The **–u** option suppresses this processing.

–r The **–r** option causes control characters to be displayed as ^C, where C stands for any such control character. Normally, **more** ignores control characters that it does not interpret in some way.

–w With **–w**, **more** prompts and waits for any key to be struck before exiting. Normally, **more** exits when it comes to the end of its input.

+linenumber
Starts up at *linenumber*.

+/pattern Starts up two lines before the line containing the regular expression *pattern*.

more uses the */usr/lib/terminfo/?* database files for the terminal characteristics and the default window size. For example, on a terminal capable of desplaying 24 lines, the default window size is 22 lines.

more looks at the environment variable, **MORE**, to preset any flags desired. For example, if you prefer to view files using the **–c** mode of operation, the shell command, **MORE=–c** in the **.profile** file causes all invocations of **more** to use this mode.

If **more** is reading from a file, rather than a pipe, a percentage is displayed along with the "--More--" prompt. This gives the fraction of the file (in characters, not lines) that has been read so far.

Other sequences which may be typed when **more** pauses, and their effects, are as follows (*i* is an optional integer argument, defaulting to 1):

i<*space*> Displays *i* more lines, (or another full screen if no arguments is given).

Ctrl-D Displays 11 more lines (scrolls). If *i* is given, then the scroll size is set to *i*.

d Same as Ctrl-D.

***i*z** Same as typing a space except that *i*, if present, becomes the new window size.

***i*s** Skips *i* lines and prints a full screen of lines.

***i*f** Skips *i* full screens and prints a full screen of lines.

***i*Ctrl-B** Skips backwards *i* full screens and prints a full screen of lines.

***i*b** Same as Ctrl-B.

q or **Q** Exits from **more**.

= Displays the current line number.

v Starts up the screen editor **vi** at the current line.

h or **?** Help command; gives a description of all the **more** commands.

i*/*expr Searches for the *i*th occurrence of the regular expression *expr*. If there are less than *i* occurrences of *expr*, and the input is a file (rather than a pipe), then the position in the file remains unchanged. Otherwise, a full screen is displayed, starting two lines before the place where the expression was found. You can use the ERASE and KILL characters to edit the regular expression. Erasing back past the first column cancels the search command.

***i*n** Searches for the *i*th occurrence of the last regular expression entered.

' (Single quotation mark) Goes to the point from which the last search started. If no search has been performed in the current file, this command goes back to the beginning of the file.

!*command*
Invokes a shell with *command*. The % and ! characters in *command* are replaced with the current filename and the previous shell command respectively. If there is no current filename, % is not expanded. The sequences, "\%" and "\!", are replaced by "%" and "!", respectively.

i:n Skips to the *i*th next file given in the command line (skips to last file if n doesn't make sense).

i:p Skips to the *i*th previous file given in the command line. If this command is given in the middle of printing out a file, **more** goes back to the beginning of the file. If *i* doesn't make sense, **more** skips back to the first file. If **more** is not reading from a file, the bell rings and nothing else happens.

:f Displays the current filename and line number.

:q or :Q Exits from **more** (same as *q* or *Q*).

. Repeats the previous command.

The commands take effect immediately; you do not need to press the RETURN key. Up to the time when the command character itself is given, you can press the KILL character to cancel the numerical argument being formed. In addition, you can press the ERASE character to redisplay the "`--More--(xx%)`" message.

The terminal is set to *noecho* mode by this program so that the output can be continuous. This means that what you type does not show on your terminal, except for the slash (/) and exclamation (!) commands.

If the standard output is not a teletype, **more** acts just like **cat**, except that a header is printed before each file, if there is more than one file.

A sample usage of **more** in previewing **nroff** output would be:

```
nroff -ms +2 doc.n | more -s
```

FILES

/usr/lib/terminfo/?/*	terminal information database
/usr/lib/more.help	help.file

SEE ALSO

ksh(1), sh(1), csh(1), environ(5)

NOTES

Before displaying a file, **more** attempts to detect whether the file is an unprintable binary file such as a directory or an executable binary image. If **more** concludes that a file is unprintable, **more** refuses to print it. However, **more** cannot detect all possible kinds of unprintable files.

NAME

mount, umount – mount and unmount filesystems

SYNOPSIS

/etc/mount [–r] [–f *fstyp*] *special directory*
/etc/mount
/etc/umount *special*
/etc/umount *directory*

DESCRIPTION

Filesystems other than **root** (/) are considered removable in the sense that they can be either available to users or unavailable. **mount** announces to the system that *special*, a block special device, is available to users from the mount point *directory*. The *directory* must already exist; it becomes the name of the root of the newly mounted *special*.

mount, when entered with arguments, adds an entry to the table of mounted devices, */etc/mnttab*. **umount** removes the entry. If invoked with no arguments, **mount** prints the entire mount table. If invoked with any of the following partial argument lists, **mount** will search */etc/fstab* to fill in the missing arguments: *special* or *directory*.

The following options are available:

–r Indicates that *special* is to be mounted read-only. If *special* is write-protected, this flag must be used.

–f *fstyp* Indicates that *fstyp* is the filesystem type to be mounted. If this argument is omitted, it defaults to the root *fstyp*.

special Indicates the block special device that is to be mounted on *directory*.

directory Indicates the directory mountpoint for *special*. (The directory must already exist.)

umount announces to the system that the previously mounted *special* is to be made unavailable. If invoked with *directory*, **umount** will search */etc/fstab* to fill in the missing argument(s).

mount can be used by any user to list mounted filesystems. Only a superuser can mount and unmount filesystems.

FILES

/etc/mnttab	Mount table
/etc/fstab	Filesystem table

SEE ALSO

fuser(1M), setmnt(1M), mount(2), umount(2), fstab(4), mnttab(4)

DIAGNOSTICS

If the **mount**(2) system call fails, **mount** prints an appropriate diagnostic. **mount** issues a warning if the filesystem to be mounted is currently labeled under another name.

umount fails if *special* is not mounted or if it is busy. *special* is busy if it contains an open file or some user's working directory. In such a case, you can use **fuser**(1M) to list and kill processes that are using *special*.

NAME

mountall, umountall – mount, unmount multiple filesystems

SYNOPSIS

/etc/mountall [–f <fstype>]
/etc/umountall [–k] [–f <fstype>]

DESCRIPTION

These commands may be executed only by the superuser.

mountall is used to mount filesystems according to the filesystem-table, */etc/fstab*.

The **–f** option will limit mounting and unmounting to the filesystem type specified. Without this option all mountable (unmountable) filesystems will be mounted (unmounted).

Before each filesystem is mounted, it is checked using **fsstat**(1M) to see if it appears mountable. Filesystems that have not been marked "clean" by a proper shutdown or **umount** will be checked, using **fsck**(1M), before the mounts are attempted. Any entry in */etc/fstab* for the root filesystem (/) is ignored.

umountall causes all mounted filesystems except **root** to be unmounted. The **–k** option sends a SIGKILL signal, via **fuser**(1M), to processes that have open files.

FILES

/etc/fstab

SEE ALSO

sysadm(1), fsck(1M), fsstat(1M), fuser(1M), mount(1M), signal(2), fstab(4)

DIAGNOSTICS

No messages are printed if the filesystems are mountable and clean, or if the filesystems are already mounted.

Error and warning messages come from **fsck**(1M), **fsstat**(1M), and **mount**(1M).

NAME

mt – magnetic tape manipulating program

SYNOPSIS

mt [**–f** *tapename*] *command* [*count*]

DESCRIPTION

mt is used to give commands to a magnetic tape drive. If a tape name is not specified, the environment variable **TAPE** is used. If **TAPE** does not exist, **mt** uses the device */dev/rmt/xt0*. Note that *tapename* must reference a raw (not block) tape device. By default, **mt** performs the requested operation once. Operations may be performed multiple times by specifying *count*.

The available commands are listed below. Only as many characters as are required to uniquely identify a command need be specified.

eof, weof Write *count* end-of-file marks at the current position on the tape.

fsf Forward space *count* files.

fsr Forward space *count* records.

bsf Back space *count* files.

bsr Back space *count* records.

rewind Rewind the tape (*count* is ignored).

offline, rewoffl Rewind the tape and place the tape unit off-line (*count* is ignored).

retension, ret Retension the tape. This command is used for streamer tapes and should be done once per data access on 5 " drives to minimize tape stretch.

eod, seod Spaces to end of valid data. Used (possibly mandatory) on streamer tape drives to append data to the logical end of tape.

noret Don't retension the tape this time. Streamer tapes are normally retensioned once on the first physical I/O operation following insertion of the tape into the drive. To disable this automatic retensioning, issue this command between inserting the tape and beginning I/O operations. Automatic tape retensioning resumes with the next tape insertion.

status Print status information about the tape unit. (This is not supported on all tape units.)

erase Erase the tape.

mt returns a 0 exit status when the operation(s) were successful, 1 if the command was unrecognized, and 2 if an operation failed.

FILES

/dev/rmt/* Raw magnetic tape interface

SEE ALSO

dd(1M), ioctl(2), environ(5), mtio(7)

BUGS

–t is a synonym for –f on the command line, for backwards compatibility.

NAME

mvdir – move a directory

SYNOPSIS

/etc/mvdir dirname name

DESCRIPTION

mvdir moves directories within a filesystem. *dirname* must be a directory. If *name* does not exist, it will be created as a directory. If *name* does exist and is a directory, *dirname* will be created as *name*/*dirname*. *dirname* and *name* may not be on the same path; that is, one may not be subordinate to the other. For example, the following command is legal:

mvdir x/y x/z

The next command is not legal:

mvdir x/y x/y/z

SEE ALSO

mkdir(1), mv(1)

WARNINGS

Only the superuser can use **mvdir**.

NAME

nawk – pattern scanning and processing language

SYNOPSIS

nawk [**–F***r*] [*parameter...*] ['*prog*'] [**–f** *progfile*] [*file...*]

DESCRIPTION

nawk is a new version of **awk** that provides capabilites unavailable in previous versions. This version will become the default version of **awk** in the next major UNIX system release.

The **–F***r* option defines the input field separator to be the regular expression *re*.

parameters in the form x=... y=... may be passed to **nawk**, where x and y are **nawk** built-in variables (see list below).

nawk scans each input *file* for lines that match any of a set of patterns specified in *prog*. The *prog* string must be enclosed in single quotes (') to protect it from the shell. For each pattern in *prog*, there may be an associated action performed when a line of a *file* matches the pattern. The set of pattern-action statements may appear literally as *prog* or in a file specified with the **–f** *progfile* option.

Input files are read in order; if there are no files, the standard input is read. The filename – means the standard input. Each input line is matched against the pattern portion of every pattern-action statement; the associated action is performed for each matched pattern.

An input line is normally made up of fields separated by white space. (This default can be changed by using the FS built-in variable or the **–F** *re* option.) The fields are denoted **$1**, **$2**, and so on; **$0** refers to the entire line.

A pattern-action statement has this form:

pattern { action }

Either pattern or action may be omitted. If there is no action with a pattern, the matching line is printed. If there is no pattern with an action, the action is performed on every input line.

Patterns are arbitrary Boolean combinations (**!**, **||**, **&&**, and parentheses) of relational expressions and regular expressions. A relational expression is one of the following:

expression relop expression
expression matchop regular expression

A relop is any of the six relational operators in C; a matchop is either **~** (contains) or **!~** (does not contain). A conditional is an arithmetic expression, a relational expression, the special expression

var **in** array,

or a Boolean combination of these.

The special patterns BEGIN and END may be used to capture control before the first input line has been read and after the last input line has been read respectively.

Regular expressions are as in **egrep** [see **grep**(1)]. In patterns, they must be surrounded by slashes. Isolated regular expressions in a pattern apply to the entire line. Regular expressions may also occur in relational expressions. A pattern may consist of two patterns separated by a comma. In this case, the action is performed for all lines between an occurrence of the first pattern and the next occurrence of the second pattern.

A regular expression may be used to separate fields by using the **-F** *re* option or by assigning the expression to the built-in variable FS. The default is to ignore leading blanks and to separate fields by blanks and/or tab characters. However, if FS is assigned a value, leading blanks are no longer ignored.

Other built-in variables include the following:

ARGC	Command line argument count
ARGV	Command line argument array
FILENAME	Name of the current input file
FNR	Ordinal number of the current record in the current file
FS	Input field separator regular expression (default blank)
NF	Number of fields in the current record
NR	Ordinal number of the current record
OFMT	Output format for numbers (default %.6g)
OFS	Output field separator (default blank)
ORS	Output record separator (default new-line)
RS	Input record separator (default new-line)

An action is a sequence of statements. A statement may be one of the following:

```
if ( conditional ) statement [ else statement ]
if ( subscript in array ) statement [ else statement ]
while ( conditional ) statement
do statement while ( conditional )
for ( expression ; conditional ; expression ) statement
for ( var in array ) statement
delete array[subscript]
break
continue
{ [ statement ] ... }
expression      # commonly variable = expression
print [ expression-list ] [ >expression ]
printf format [ , expression-list ] [ >expression ]
next            # skip remaining patterns on this input line
exit [expr]     # skip the rest of the input; exit status is expr
return [expr]
```

Statements are terminated by semicolons, new-lines, or right braces. An empty expression-list stands for the whole input line. Expressions take on string or numeric values as appropriate, and are built using the operators +, -, *, /, %, and concatenation (indicated by a blank). The C operators ++, —, +=,

-=, *=, /=, and %= are also available in expressions. Variables may be scalars, array elements (denoted x[i]), or fields. Variables are initialized to the null string or zero. Array subscripts may be any string, not necessarily numeric; this allows for a form of associative memory. String constants are quoted (").

The **print** statement prints its arguments on the standard output, or on a file if *>expression* is present, or on a pipe if | *cmd* is present. The arguments are separated by the current output field separator and terminated by the output record separator. The **printf** statement formats its expression list according to the format [see **printf**(3S)].

nawk has a variety of built-in functions: arithmetic, string, input/output, and general.

The arithmetic functions are *atan2*, *cos*, *exp*, *int*, *log*, *rand*, *sin*, *sqrt*, and *srand*. *int* truncates its argument to an integer. *rand* returns a random number between 0 and 1. *srand* (*expr*) sets the seed value for *rand* to *expr* or uses the time of day if *expr* is omitted.

The string functions are as follows:

gsub(for, repl, in) Behaves like *sub* (see below), except that it replaces successive occurrences of the regular expression (like the **ed** global substitute command).

index(s, t) Returns the position in string *s* where string *t* first occurs, or 0 if it does not occur at all.

length(s) Returns the length of its argument taken as a string, or of the whole line if there is no argument.

match(s, re) Returns the position in string *s* where the regular expression *re* occurs, or 0 if it does not occur at all. RSTART is set to the starting position (which is the same as the returned value), and RLENGTH is set to the length of the matched string.

split(s, a, fs) Splits the string *s* into array elements *a*[*1*], *a*[*2*], *a*[*n*], and returns *n*. The separation is done with the regular expression *fs* or with the field separator FS if *fs* is not given.

sprintf(fmt, expr, expr, ...) Formats the expressions according to the **printf**(3S) format given by *fmt* and returns the resulting string.

sub(for, repl, in) Substitutes the string *repl* in place of the first instance of the regular expression *for* in string *in* and returns the number of substitutions. If *in* is omitted, **nawk** substitutes in the current record ($0).

substr(s, m, n) Returns the *n*-character substring of *s* that begins at position *m*.

The input/output and general functions are as follows:

close(filename) Closes the file or pipe named *filename*.

cmd | getline Pipes the output of *cmd* into *getline*; each successive call to *getline* returns the next line of output from *cmd*.

getline	Sets **$0** to the next input record from the current input file.
getline <file	Sets **$0** to the next record from *file*.
getline var	Sets variable *var* instead.
getline var <file	Sets *var* from the next record of *file*.
system(cmd)	Executes *cmd* and returns its exit status.

All forms of *getline* return 1 for successful input, 0 for end of file, and –1 for an error.

nawk also provides user-defined functions. Such functions may be defined (in the pattern position of a pattern-action statement) as follows:

function name(args,...) { stmts }
func name(args,...) { stmts }

Function arguments are passed by value if scalar, and by reference if array name. Argument names are local to the function; all other variable names are global. Function calls may be nested and functions may be recursive. The **return** statement may be used to return a value.

EXAMPLES

Print lines longer than 72 characters:

```
length > 72
```

Print first two fields in opposite order:

```
{ print $2, $1 }
```

Same, with input fields separated by comma and/or blanks and tabs:

```
BEGIN { FS = ",[ \t]*|[ \t]+" }
      { print $2, $1 }
```

Add up first column, print sum and average:

```
      { s += $1 }
END   { print "sum is", s, " average is", s/NR }
```

Print fields in reverse order:

```
{ for (i = NF; i > 0; --i) print $i }
```

Print all lines between start/stop pairs:

```
/start/, /stop/
```

Print all lines whose first field is different from previous one:

```
$1 != prev { print; prev = $1 }
```

Simulate **echo**(1):

```
BEGIN {
      for (i = 1; i < ARGC; i++)
            printf "%s", ARGV[i]
      printf "\n"
      exit
      }
```

Print file, filling in page numbers starting at 5:

```
/Page/  {  $2  =  n++;  }
          {  print  }
```

command line: **nawk –f program n=5 input**

SEE ALSO

awk(1), grep(1), lex(1), sed(1), printf(3S)

BUGS

Input white space is not preserved on output if fields are involved.

There are no explicit conversions between numbers and strings. To force an expression to be treated as a number, add 0 to it. To force an expression to be treated as a string, concatenate the null string ("") to it.

NAME

ncheck – generate pathnames from i-numbers

SYNOPSIS

/etc/ncheck [**–i** i-numbers] [**–a**] [**–s**] [filesystem]

DESCRIPTION

ncheck with no arguments generates a pathname vs. i-number list of all files on a set of default filesystems (see */etc/checklist*). Names of directory files are followed by **/..**

The options are as follows:

–i Limits the report to only those files whose i-numbers follow.

–a Allows printing of the names . and .., which are ordinarily suppressed.

–s Limits the report to special files and files with set-user-ID mode. This option may be used to detect violations of security policy.

filesystem must be specified by the fileystem's special file.

The report should be sorted so that it is more useful.

SEE ALSO

sort(1), fsck(1M)

DIAGNOSTICS

If the filesystem structure is not consistent, **??** denotes the "parent" of a parentless file and a pathname beginning with **...** denotes a loop.

NAME

newform – change the format of a text file

SYNOPSIS

newform [**–s**] [**–i***tabspec*] [**–o***tabspec*] [**–b***n*] [**–e***n*] [**–p***n*] [**–a***n*] [**–f**] [**–c***char*] [**–l***n*] [*files*]

DESCRIPTION

newform reads lines from the named *files* (or from the standard input if no input file is named) and reproduces the lines on the standard output. Lines are reformatted in accordance with command line options in effect.

Except for **–s**, command line options may appear in any order, may be repeated, and may be intermingled with the optional *files*. Command line options are processed in the order specified. This means that option sequences like **–e**15 **–l**60 will yield results different from **–l**60 **–e**15. Options are applied to all *files* on the command line.

–s Shears off leading characters on each line up to the first tab and places up to 8 of the sheared characters at the end of the line. If more than 8 characters (not counting the first tab) are sheared, the eighth character is replaced by a * and any characters to the right of it are discarded. The first tab is always discarded.

An error message and program exit occurs if this option is used on a file without a tab on each line. The characters sheared off are saved internally until all other options specified are applied to that line. The characters are then added at the end of the processed line.

For example, to convert a file with leading digits, one or more tabs, and text on each line, to a file beginning with the text, all tabs after the first expanded to spaces, padded with spaces out to column 72 (or truncated to column 72), and the leading digits placed starting at column 73, the command would be as follows:

newform –s –i –l –a –e *filename*

–i*tabspec* Input tab specification: expands tabs to spaces, according to the tab specifications given. *tabspec* recognizes all tab specification forms described in **tabs**(1). In addition, *tabspec* can be **––**, in which **newform** assumes that the tab specification is to be found in the first line read from the standard input (see **fspec**(4)). If no *tabspec* is given, *tabspec* defaults to **–8**. A *tabspec* of **–0** expects no tabs; if any are found, they are treated as **–1**.

–o*tabspec* Output tab specification: replaces spaces by tabs, according to the tab specifications given. The tab specifications are the same as for **–i***tabspec*. If no *tabspec* is given, *tabspec* defaults to **–8**. A *tabspec* of **–0** means that no spaces will be converted to tabs on output.

–b*n* Truncate *n* characters from the beginning of the line when the line length is greater than the effective line length (see **–l***n*). Default is to truncate the number of characters necessary to obtain the effective line length. The default value is used when **–b** with no *n* is used. This option can be used to delete the sequence numbers from a COBOL program as follows:

newform -l1 -b7 *filename*

-e*n* Same as **-b***n* except that characters are truncated from the end of the line.

-p*n* Prefix *n* characters (see **-c***k*) to the beginning of a line when the line length is less than the effective line length. Default is to prefix the number of characters necessary to obtain the effective line length.

-a*n* Same as **-p***n* except characters are appended to the end of a line.

-f Write the tab specification format line on the standard output before any other lines are output. The tab specification format line that is printed will correspond to the format specified in the *last* **-o** option. If no **-o** option is specified, the line that is printed will contain the default specification of **-8**.

-c*k* Change the prefix/append character to *k*. Default character for *k* is a space.

-l*n* Set the effective line length to *n* characters. If *n* is not entered, **-l** defaults to 72. The default line length without the **-l** option is 80 characters. Note that tabs and backspaces are considered to be one character (use **-i** to expand tabs to spaces).

The **-l1** option must be used to set the effective line length shorter than any existing line in the file so that the **-b** option is activated.

SEE ALSO

csplit(1), tabs(1), fspec(4)

DIAGNOSTICS

All diagnostics are fatal.

usage: ...	**newform** was called with a bad option.
not -s format	There was no tab on one line.
can't open file	Self-explanatory.
internal line too long	A line exceeds 512 characters after being expanded in the internal work buffer.
tabspec in error	A tab specification is incorrectly formatted, or specified tab stops are not ascending.
tabspec indirection illegal	A *tabspec* read from a file (or standard input) may not contain a *tabspec* referencing another file (or standard input).

0 – normal execution; 1 – for any error

BUGS

newform normally keeps track of physical characters only. However, for the **-i** and **-o** options, **newform** will keep track of backspaces in order to line up tabs in the appropriate logical columns. **newform** will not prompt the user if a *tabspec* is to be read from the standard input (by use of **-i--** or **-o--**).

If the **-f** option is used, the last **-o** option specified was **-o--**, and it was preceded by either a **-o--** or a **-i--**, the tab specification format line will be incorrect.

NAME

newfs – construct a new file system

SYNOPSIS

/etc/newfs [**–N**] [[**–v**] [options] **special**

DESCRIPTION

The operating system programs **newfs** and **mkfs**(1M) are the same program binary. **newfs** provides a friendly interface to create file systems on disks with a VTOC. *special* is the device special file on which the new filesystem is to be created, and may be specified with no pathname */dev/rdsk* prefix. **newfs** consults the VTOC residing on the disk to determine the characteristics of the disk so it can calculate the appropriate parameters to use in constructing a file system on that disk.

Note that if a VTOC is not present on the disk, **newfs** cannot create a filesystem on that disk. (Use **mkvtoc**(1M) to add a VTOC.)

If the **–v** option is specified, **newfs** prints out parameters which could be supplied to the **mkfs** interface for making file systems.

The **–N** option causes the file system parameters to be printed out without really creating the file system.

Note that **newfs** will not install any system bootstraps on the disk. To install a bootstrap, use the **installboot**(1M) command.

newfs uses **fsirand**(1M) as a security precaution for file systems which are mounted over NFS.

The following options define the general layout policies.

–b *blocksize*
: The block size of the file system in bytes. The default is taken from the VTOC.

–f *frag_size*
: The fragment size of the file system in bytes. The default is taken from the VTOC.

–m *free_space %*
: The percentage of space reserved from normal users; the minimum free space threshold. The default is 10%. See **tunefs**(1M) for more information on setting this option.

–a *maxcontig*
: The maximum number of contiguous blocks that will be laid out before forcing a rotational delay (see **–d** below). The default is 1. See **tunefs**(1M) for more information on setting this option.

–d *rotdelay*
: The expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. The default is 4 milliseconds. See **tunefs**(1M) for more information on setting this option.

–e *maxbpg*
: Indicates the maximum number of blocks any single file can allocate from one cylinder group before it is forced to begin allocating

blocks from another cylinder group. The default is about one quarter of the total blocks in a cylinder group. See **tunefs**(1M) for more information on setting this option.

–i *number_of_bytes_per_inode*
This specifies the density of inodes in the file system. The default is to create an inode for each 2048 bytes of data space. If fewer inodes are desired, a larger number should be used; to create more inodes a smaller number should be given. Be sure to specify this option for small filesystems (less than one cylinder group) or the number of inodes created will be too large.

–c *#cylinders_per_group*
The number of cylinders per cylinder group in a file system. The default is 16.

–s *size* The size of the file system in sectors. The default is taken from the VTOC.

The following options override the standard sizes for the disk geometry. Their default values are taken from the VTOC, or are coded internally to **newfs**. Changing these defaults is useful when using **newfs** to build a file system whose raw image will eventually be used on a different type of disk than the one on which it is initially created (for example on a write-once disk).

–r *revolutions_per_minute*
The speed of the disk in revolutions per minute.

–S *sector-size*
The size of a sector in bytes (almost never anything but 512).

–u *sectors_per_track*
The number of sectors/track available for data allocation by the file system. This does not include sectors reserved at the end of each track for bad block replacement (see **–p** below).

–t *#tracks_per_cylinder*
The number of tracks/cylinder available for data allocation by the file system.

–p *spare_sectors_per_track*
Spare sectors (bad sector replacements) are physical sectors that occupy space at the end of each track. They are not counted as part of the sectors/track (**–u**) since they are not available to the file system for data allocation.

–x *spare_sectors_per_cylinder*
Spare sectors (bad sector replacements) are physical sectors that occupy space at the end of the last track in the cylinder. They are deducted from the sectors/track (**–u**) of the last track of each cylinder since they are not available to the file system for data allocation.

–l *hardware_sector_interleave*
Used to describe the media format to compensate for a slow controller. Interleave is physical sector interleave on each track, specified as the denominator of the ratio:

sectors read / sectors passed over

Thus an interleave of 1/1 implies contiguous layout, while 1/2 implies logical sector 0 is separated by one sector from logical sector 1.

-k *skew* Used to describe the media format to compensate for a slow controller. Track skew is the offset of sector 0 on track N relative to sector 0 on track N-1 on the same cylinder.

SEE ALSO

devbuild(1M), fsck(1M), fsirand(1M), installboot(1M), mkfs(1M), mkvtoc(1M), prtvtoc(1M), tunefs(1M), fs(4)

NAME

newgrp – log in to a new group

SYNOPSIS

newgrp [–] [group]

DESCRIPTION

newgrp changes a user's group identification. The user remains logged in and the current directory is unchanged, but calculations of access permissions to files are performed with respect to the new real and effective group IDs. The user is always given a new shell, replacing the current shell, by **newgrp**, regardless of whether it terminated successfully or due to an error condition (i.e., unknown group).

Exported variables retain their values after invoking **newgrp**; however, all unexported variables are either reset to their default value or set to null. System variables (such as **PS1**, **PS2**, **PATH**, **MAIL**, and **HOME**), unless exported by the system or explicitly exported by the user, are reset to default values. For example, a user has a primary prompt string (**PS1**) other than **$** (default) and has not exported **PS1**. After an invocation of **newgrp**, successful or not, their **PS1** will now be set to the default prompt string **$**. Note that the shell command **export** (see **sh**(1)) is the method to export variables so that they retain their assigned value when invoking new shells.

With no arguments, **newgrp** changes the group identification back to the group specified in the user's password file entry. This is a way to exit the effect of an earlier **newgrp** command.

If the first argument to **newgrp** is a –, the environment is changed to what would be expected if the user actually logged in again as a member of the new group.

A password is demanded if the group has a password and the user does not, or if the group has a password and the user is not listed in */etc/group* as being a member of that group.

FILES

/etc/group	System group file
/etc/passwd	System password file

SEE ALSO

login(1), sh(1), group(4), passwd(4), environ(5)

BUGS

There is no convenient way to enter a password into */etc/group*. Use of group passwords is not encouraged, because, by their very nature, they encourage poor security practices. Group passwords may disappear in the future.

NAME

news – print news items

SYNOPSIS

news [**–a**] [**–n**] [**–s**] [items]

DESCRIPTION

news is used to keep the user informed of current events. By convention, these events are described by files in the directory */usr/news*.

When invoked without arguments, **news** prints the contents of all current files in */usr/news*, most recent first, with each preceded by an appropriate header. **news** stores the "currency" time as the modification date of a file named *.news_time* in the user's home directory (the identity of this directory is determined by the environment variable **$HOME**). Only files more recent than this currency time are considered "current."

The options to the **news** command are as follows:

–a Causes **news** to print all items, regardless of currency. In this case, the stored time is not changed.

–n Causes **news** to report the names of the current items without printing their contents and without changing the stored time.

–s Causes **news** to report how many current items exist, without printing their names or contents, and without changing the stored time. It is useful to include such an invocation of **news** in one's *.profile* file or in the system */etc/profile*.

All other arguments are assumed to be specific news items that are to be printed.

If the Delete key is typed during the printing of a news item, printing stops and the next item is started. Another Delete within one second of the first causes the program to terminate.

FILES

/etc/profile
/usr/news/*
$HOME/.news_time

SEE ALSO

profile(4), environ(5)

NAME

nice – run a command at low priority

SYNOPSIS

nice [–increment] command [arguments]

DESCRIPTION

nice executes *command* with a lower CPU scheduling priority. If an *increment* argument (in the range 1-19) is given, it is used. If not, an increment of 10 is assumed.

The superuser may run commands with priority higher than normal by using a negative increment such as **––10**.

SEE ALSO

nohup(1), sh(1), nice(2)

DIAGNOSTICS

nice returns the exit status of the subject command.

BUGS

An *increment* larger than 19 is equivalent to 19.

NAME

nl – line numbering filter

SYNOPSIS

nl [**–h***type*] [**–b***type*] [**–f***type*] [**–v***start#*] [**–i***incr*] [**–p**] [**–l***num*] [**–s***sep*] [**–w***width*] [**–n***format*] [**–d***delim*] file

DESCRIPTION

nl reads lines from the named *file* (or from the standard input if no *file* is named) and reproduces the lines on the standard output. Lines are numbered on the left in accordance with the command options in effect.

nl views the text it reads in terms of logical pages. Line numbering is reset at the start of each logical page. A logical page consists of a header, a body, and a footer section. Empty sections are valid. Different line numbering options are independently available for header, body, and footer (for example, no numbering of header and footer lines while numbering blank lines only in the body).

The start of logical page sections are signaled by input lines containing nothing but the following delimiter character(s):

Line contents	*Start of*
\:\:\:	header
\:\:	body
\:	footer

Unless optioned otherwise, **nl** assumes the text being read is in a single logical page body.

Command options may appear in any order and may be intermingled with an optional filename. Only one file may be named. The options are as follows:

–b*type* Specifies which logical page body lines are to be numbered. Recognized *types* and their meanings are as follows:

a	Number all lines
t	Number lines with printable text only
n	No line numbering
p*string*	Number only lines containing the regular expression specified in *string*

–h*type* Same as **–b***type* except for header. Default *type* for logical page header is **n** (no lines numbered). Default *type* for logical page body is **t** (text lines numbered).

–f*type* Same as **–b***type* except for footer. Default for logical page footer is **n** (no lines numbered).

–v*start#* *start#* is the initial value used to number logical page lines. Default is **1**.

–i*incr* *incr* is the increment value used to number logical page lines. Default is **1**.

–p Do not restart numbering at logical page delimiters.

-l*num* *num* is the number of blank lines to be considered as one. For example, **-l2** results in only the second adjacent blank being numbered (if the appropriate **-ha**, **-ba**, and/or **-fa** option is set). Default is **1**.

-s*sep* *sep* is the character(s) used in separating the line number and the corresponding text line. Default *sep* is a tab.

-w*width* *width* is the number of characters to be used for the line number. Default *width* is **6**.

-n*format* *format* is the line numbering format. Recognized values are: **ln** (left justified, leading zeros suppressed), **rn** (right justified, leading zeros suppressed), **rz** (right justified, leading zeros kept). Default *format* is **rn** (right justified).

-d*xx* The delimiter characters specifying the start of a logical page section may be changed from the default characters (\:) to two user-specified characters. If only one character is entered, the second character remains the default character (:). No space should appear between the **-d** and the delimiter characters. To enter a backslash, use two backslashes.

EXAMPLES

The following command will number *file1* starting at line number 10 with an increment of 10.

```
nl -v10 -i10 -d!+ file1
```

The logical page delimiters are !+.

SEE ALSO

pr(1)

NAME

noage, nopff, noswap, onproc – run a command with priority aging, PFF, swapping, or processor migration disabled

SYNOPSIS

/etc/noage command [arguments]

/etc/nopff command [arguments]

/etc/noswap command [arguments]

/etc/onproc proc command [arguments]

DESCRIPTION

noage executes *command* (and its children) with priority aging disabled. Ordinarily, process priority is an inverse function of the process's CPU usage in the recent past. (See **proc_ctl**(2)).

nopff executes *command* without the command (or its children) being subjected to PFF adjustments. PFF (Page Fault Frequency) adjustments are changes in the resident-set size based on the frequency of page faults. (See **vm_ctl**(2)).

noswap executes *command* with swapping for the command (and its children) disabled. (See **vm_ctl**(2)).

onproc executes *command* on processor number *proc*, with processor migration disabled for the command (and its children). (See **tmp_affinity**(2)).

These commands may be nested. For example, the following line executes *mycommand* with swapping, PFF-ing, and processor migration disabled.

/etc/nopff /etc/noswap /etc/onproc *0 mycommand arg1 arg2*

These commands are effective only when invoked by the superuser, unless the underlying system call option has been configured for non-superuser processes. See the **proc_ctl**(2) and **vm_ctl**(2) manual entries for configuration options.

SEE ALSO

proc_ctl(2), tmp_affinity(2), vm_ctl(2)

NAME

nohup – run a command immune to hangups and quits

SYNOPSIS

nohup command [arguments]

DESCRIPTION

nohup executes *command* with hangups and quits ignored. If output is not re-directed by the user, both standard output and standard error are sent to *nohup.out*. If *nohup.out* is not writable in the current directory, output is redirected to *$HOME/nohup.out*.

EXAMPLES

It is frequently desirable to apply **nohup** to pipelines or lists of commands. This can be done only by placing pipelines and command lists in a single file, called a shell procedure. When you issue the following command, **nohup** applies to everything in *file*.

nohup sh *file*

If the shell procedure *file* is to be executed often, then the need to type **sh** can be eliminated by giving *file* execute permission. Add an ampersand and the contents of *file* are run in the background with interrupts also ignored (see **sh**(1)):

nohup file &

Following is an example of the contents of *file*:

sort ofile > nfile

SEE ALSO

chmod(1), nice(1), sh(1), signal(2)

WARNINGS

In the following command, **nohup** applies only to command1.

nohup command1;command2

The following command is syntactically incorrect.

nohup (command1;command2)

NAME

od – octal dump

SYNOPSIS

od [**–bcdosx**] [file] [[**+**]offset[**.**][**b**]]

DESCRIPTION

od dumps *file* in one or more formats as selected by the first argument. If the first argument is missing, **–o** is default. The meanings of the format options are as follows:

–b Interpret bytes in octal.

–c Interpret bytes in ASCII. Certain non-graphic characters appear as C escapes: null=**\0**, backspace=**\b**, form-feed=**\f**, newline=**\n**, return=**\r**, tab=**\t**; others appear as 3-digit octal numbers.

–d Interpret words in unsigned decimal.

–o Interpret words in octal.

–s Interpret 16-bit words in signed decimal.

–x Interpret words in hex.

The *file* argument specifies which file is to be dumped. If no file argument is specified, the standard input is used.

The offset argument specifies the offset in the file where dumping is to commence. This argument is normally interpreted as octal bytes. If a dot (.) is appended, the offset is interpreted in decimal. If **b** is appended, the offset is interpreted in blocks of 512 bytes. If the file argument is omitted, the offset argument must be preceded by +.

Dumping continues until end-of-file.

NAME

online, offline – bring processors online and offline

SYNOPSIS

/etc/online –N

/etc/online –S

/etc/online [**–a**] [**–v**] [processor-number ...]

/etc/offline [**–a**] [**–v**] [processor-number ...]

DESCRIPTION

online –N causes the number of currently online processes to be printed. This is very useful in shell scripts.

online –S causes a one line summary on the online processors to be printed. Also very useful in shell scripts.

online causes the given processors to initialize and begin scheduling user processes. Processors are numbered consecutively upward from zero. If the **–a** option is present, all processors in the system are brought online.

offline removes the given processor[s] from the pool of processors available for work. One processor is always left online. If the **–a** option is present, all but one of the processors in the system are brought offline.

Both **online** and **offline** display which processors are on and offline. If the **–v** option is specified, any resulting changes in state (on or offline) are also displayed.

Only the superuser can take processors on or offline.

SEE ALSO

shutdown(1M), tmp_ctl(2), tmp_affinity(2)

NAME

pack, pcat, unpack – compress and expand files

SYNOPSIS

pack [–] [**–f**] name ...

pcat name ...

unpack name ...

DESCRIPTION

pack attempts to store the specified files in a compressed form. Wherever possible (and useful), each input file *name* is replaced by a packed file *name.z* with the same access modes, access and modified dates as those of *name*. The **–f** option will force packing of *name*. This is useful for causing an entire directory to be packed even if some of the files will not benefit. If **pack** is successful, *name* will be removed. Packed files can be restored to their original form by using **unpack** or **pcat**.

pack uses Huffman (minimum redundancy) codes on a byte-by-byte basis. If the – argument is used, an internal flag is set that causes the number of times each byte is used, its relative frequency, and the code for the byte to be printed on the standard output. Additional occurrences of – in place of *name* will cause the internal flag to be set and reset.

The amount of compression obtained depends on the size of the input file and the character frequency distribution. Because a decoding tree forms the first part of each .z file, it is usually not worthwhile to pack files smaller than three blocks unless the character frequency distribution is very skewed, which may occur with printer plots or pictures.

Typically, text files are reduced to 60-75% of their original size. Load modules, which use a larger character set and have a more uniform distribution of characters, show little compression, the packed versions being about 90% of the original size.

pack returns a value that is the number of files that it failed to compress.

No packing will occur if any of the following conditions exist:

The file appears to be already packed;
The file has links;
The file is a directory;
The file cannot be opened;
No disk storage blocks will be saved by packing;
A file called *name.z* already exists;
The .z file cannot be created;
An I/O error occurred during processing.

Directories cannot be compressed.

pcat does for packed files what **cat**(1) does for ordinary files, except that **pcat** cannot be used as a filter. The specified files are unpacked and written to the standard output. Thus, to view a packed file named *name.z*, you could use this command:

pcat name.z

or just:

pcat name

To make an unpacked copy, say *nnn*, of a packed file named *name.z* (without destroying *name.z*) use this command:

pcat name >nnn

pcat returns the number of files it was unable to unpack. Failure may occur if any of these conditions exist:

The file cannot be opened;
The file does not appear to be the output of **pack**.

Unpack expands files created by **pack**. For each file *name* specified in the command, a search is made for a file called *name.z* (or just *name*, if *name* ends in *.z*). If this file appears to be a packed file, it is replaced by its expanded version. The new file has the *.z* suffix stripped from its name and has the same access modes, access and modification dates as those of the packed file.

unpack returns a value that is the number of files it was unable to unpack. Failure may occur for the same reasons that it may in **pcat**, as well as for the following:

A file with the "unpacked" name already exists;
The unpacked file cannot be created.

SEE ALSO

cat(1)

NAME

passmgmt – password files management

SYNOPSIS

passmgmt –a *options name*
passmgmt –m *options name*
passmgmt –d *name*

DESCRIPTION

The **passmgmt** command updates information in the password files. This command works with both */etc/passwd* and */etc/shadow*. If there is no */etc/shadow*, the changes done by **passmgmt** will go only to */etc/passwd*.

passmgmt –a adds an entry for user *name* to the password files. This command does not create any directory for the new user and the new login remains locked (with the string `*LK*` in the password field) until the **passwd**(1) command is executed to set the password.

passmgmt –m modifies the entry for user *name* in the password files. The name field in the */etc/shadow* entry and all fields (except the password field) in the */etc/passwd* entry can be modified by this command. Only fields entered on the command line will be modified.

passmgmt –d deletes the entry for user *name* from the password files. It will not remove any files that the user owns on the system; they must be removed manually.

The following additional options are available:

–c *comment* A short description of the login. It is limited to a maximum of 128 characters and defaults to an empty field.

–h *homedir* Home directory of *name*. It is limited to a maximum of 256 characters and defaults to */usr/name*.

–u *uid* UID of the *name*. This number must range from 0 to the maximum non-negative value for the system. It defaults to the next available UID greater than 99. Without the **–o** option, it enforces the uniqueness of a UID.

–o This option allows a UID to be non-unique. It is used only with the **–u** option.

–g *gid* GID of the *name*. This number must range from 0 to the maximum non-negative value for the system. The default is 1.

–s *shell* Login shell for *name*. It should be the full pathname of the program that will be executed when the user logs in. The maximum size of *shell* is 256 characters. The default value is for this field to be empty and to be interpreted as **/bin/sh**.

–l *logname* This option changes the *name* to *logname*. It is used only with the **–m** option.

–p *passwd* Encrypted password for *name*. This option inserts the already encrypted *passwd* string into the proper password file instead of the '*LK*' string. The *passwd* string should consist entirely of characters in the regular expression [./0-9A-Za-z] and must be exactly 13 characters long. This option should be used with

caution because there is no way to determine the corresponding string that must be entered by a user to achieve *passwd*.

The total size of each login entry is limited to a maximum of 511 bytes in each of the password files.

FILES

/etc/passwd, /etc/shadow, /etc/opasswd, /etc/oshadow

SEE ALSO

passwd(1), passwd(4)

DIAGNOSTICS

The **passmgmt** command exits with one of the following values:

0	Success.
1	Permission denied.
2	Invalid command syntax. Usage message of the **passmgmt** command will be displayed.
3	Invalid argument provided to option.
4	UID in use.
5	Inconsistent password files (for example, *name* is in the */etc/passwd* file and not in the */etc/shadow* file, or vice versa).
6	Unexpected failure. Password files unchanged.
7	Unexpected failure. Password file(s) missing.
8	Password file(s) busy. Try again later.
9	*name* does not exist (if **–m** or **–d** is specified), already exists (if **–a** is specified), or *logname* already exists (if **–m –l** is specified).

NOTES

You cannot use a colon or <Return> as part of an argument because it will be interpreted as a field separator in the password file.

NAME

passwd – change login password and password attributes

SYNOPSIS

passwd [*name*]

passwd [**-l** | **-d**] [**-n** *min*] [**-f**] [**-x** *max*] *name*

passwd -s [**-a**]

passwd -s [*name*]

DESCRIPTION

The **passwd** command changes a user's password or lists password attributes associated with the user's login *name*. Superusers can use **passwd** to install or change passwords and attributes associated with any login *name*.

When used to change a password, **passwd** prompts ordinary users for their old password, if any. It then prompts for the new password twice. When the old password is entered, **passwd** checks to see if it has aged sufficiently. If aging is insufficient, **passwd** terminates; see **passwd**(4).

Assuming aging is sufficient, a check is made to ensure that the new password meets construction requirements. When the new password is entered a second time, the two copies of the new password are compared. If the two copies are not identical, the cycle of prompting for the new password is repeated for at most two more times.

Passwords must be constructed to meet the following requirements:

Each password must have at least 6 characters. Only the first 8 characters are significant.

Each password must contain at least 2 alphabetic characters and at least 1 numeric or special character. In this case, *alphabetic* refers to all upper- or lowercase letters.

Each password must differ from the user's login *name* and any reverse or circular shift of that login *name*. For comparison purposes, an uppercase letter and its corresponding lowercase letter are equivalent.

New passwords must differ from the old by at least 3 characters. For comparison purposes, an uppercase letter and its corresponding lowercase letter are equivalent.

Superusers (that is, real and effective uid equal to zero, see **id**(1M) and **su**(1M)) may change any password; hence, **passwd** does not prompt superusers for the old password. Superusers are not forced to comply with password aging and password construction requirements. A superuser can create a null password by entering a carriage return in response to the prompt for a new password. (This differs from **passwd -d** because the "password" prompt will still be displayed.)

Any user can use the **-s** option to show password attributes for his or her own login *name*. The format of the display is as follows:

name status mm/dd/yy min max

If password aging information is not present, the display has this format:

name status

The format has this meaning:

name The login ID of the user.

status The password status of *name*: PS stands for passworded or locked, LK stands for locked, and NP stands for no password.

mm/dd/yy The date the password was last changed for *name*. (Note that all password aging dates are determined using Greenwich Mean Time and, therefore, may differ by as much as a day in other time zones.)

min The minimum number of days required between password changes for *name*.

max The maximum number of days the password is valid for *name*.

Only a superuser can use the following options:

-l Locks password entry for *name*.

-d Deletes password for *name*. The login *name* will not be prompted for password.

-n Set minimum field for *name*. The *min* field contains the minimum number of days between password changes for *name*. If *min* is greater than *max*, the user may not change the password. Always use this option with the **-x** option, unless *max* is set to -1 (aging turned off). In that case, *min* need not be set.

-x Set maximum field for *name*. The *max* field contains the number of days that the password is valid for *name*. The aging for *name* will be turned off immediately if *max* is set to -1. If it is set to 0, then the user is forced to change the password at the next login session and aging is turned off.

-a Show password attributes for all entries. Use only with **-s** option; *name* must not be provided.

-f Force the user to change password at the next login by expiring the password for *name*.

FILES

/etc/passwd, /etc/shadow, /etc/opasswd, /etc/oshadow

SEE ALSO

login(1), id(1M), passmgmt(1M), pwconv(1M), su(1M), crypt(3C), passwd(4)

DIAGNOSTICS

The **passwd** command exits with one of the following values:

0 Success.

1 Permission denied.

2 Invalid combination of options.

3 Unexpected failure. Password file unchanged.

4 Unexpected failure. Password file(s) missing.

5 Password file(s) busy. Try again later.

6 Invalid argument to option.

WARNING

If the optional */etc/shadow* file feature is used, the **passwd**(1) command will use that file, instead of the */etc/passwd* file, to obtain password information. Because password aging information is stored in the two files in a slightly different manner, the output from **passwd** options that use this information may also be different.

NAME

paste – merge same lines of several files or subsequent lines of one file

SYNOPSIS

paste file1 file2 ...
paste –dlist file1 file2 ...
paste –s [**–d**list] file1 file2 ...

DESCRIPTION

In the first two forms, **paste** concatenates corresponding lines of the given input files *file1*, *file2*, and so on. It treats each file as a column or columns of a table and pastes them together horizontally (parallel merging). It is the counterpart of **cat**(1), which concatenates vertically (one file after the other). In the last form above, **paste** replaces the function of an older command with the same name by combining subsequent lines of the input file (serial merging). In all cases, lines are glued together with the Tab character or with characters from an optionally specified *list*. Output is to the standard output and can be used as the start of a pipe or as a filter, if – is used in place of a filename.

The meanings of the options are as follows:

–d Without this option, the newline characters of each but the last file (or last line in case of the **–s** option) are replaced by a Tab character. This option allows replacing the Tab character by one or more alternate characters (see below).

list One or more characters immediately following **–d** replace the default *tab* as the line concatenation character. The list is used circularly; when exhausted, it is reused. In parallel merging (no **–s** option), the lines from the last file are always terminated with a newline character, not from the *list*. The list may contain these special escape sequences: **\n** (newline), **\t** (tab), **** (backslash), and **\0** (empty string, not a null character). Quoting may be necessary if characters have special meaning to the shell (for example, to get one backslash, use **–d"\\\\"**).

–s Merge subsequent lines rather than one from each input file. Use Tab for concatenation, unless a *list* is specified with the **–d** option. Regardless of the *list*, the last character of the file is forced to be a newline.

– May be used in place of any filename, to read a line from the standard input. (There is no prompting).

EXAMPLES

ls | paste –d"" –	List directory in one column
ls | paste – – – –	List directory in four columns
paste –s –d"\t\n" *file*	Combine pairs of lines into lines

SEE ALSO

cut(1), grep(1), pr(1)

DIAGNOSTICS

line too long	Output lines are restricted to 511 characters.
too many files	Except for **–s** option, no more than 12 input files may be specified.

NAME

pathconv – search Interpreter criteria for filename

SYNOPSIS

pathconv [**–v** alias] [**–f**] [**–t**]

DESCRIPTION

The **pathconv** command is used to convert an alias to its pathname. It takes the alias as a string from *stdin*. The **–v** option allows the alias to be embedded in the command string. The **–t** option implies that **pathconv** should expand the alias in a format suitable for display as a frame title. This format is a shortened version of the full pathname, created by deleting components of the path from the middle of the string until it is under 42 characters in length, and then inserting "..." between the remaining pieces. The **–f** option means return the full path (this is the default).

EXAMPLES

Here is a menu that is titled using **pathconv**:

```
menu=`pathconv -t -v $ARG1`
    .
    .
    .
```

This will result in the same thing:

```
Menu=`echo $ARG1 | pathconv -t`
    .
    .
    .
```

SEE ALSO

echo(1F)

NAME

pg – file perusal filter for CRTs

SYNOPSIS

pg [*–number*] [**–p** *string*] [**–cefns**] [*+linenumber*] [*+/pattern/*] [files...]

DESCRIPTION

The **pg** command is a filter that allows the examination of *files* one screenful at a time on a CRT. (The filename – and/or NULL arguments indicate that **pg** should read from the standard input.) Each screenful is followed by a prompt. If the user types a carriage return, another page is displayed; other possibilities are enumerated below.

This command is different from previous paginators in that it allows you to back up and review something that has already passed. The method for doing this is explained below.

In order to determine terminal attributes, **pg** scans the **terminfo**(4) database for the terminal type specified by the environment variable **TERM**. If this variable is not defined, the terminal type **dumb** is assumed.

The command line options are as follows:

–number An integer specifying the size (in lines) of the window that **pg** is to use instead of the default. (On a terminal containing 24 lines, the default window size is 23).

–p *string* Causes **pg** to use *string* as the prompt. If the prompt string contains a "%d", the first occurrence of "%d" in the prompt will be replaced by the current page number when the prompt is issued. The default prompt string is ":".

–c Home the cursor and clear the screen before displaying each page. This option is ignored if **clear_screen** is not defined for this terminal type in the **terminfo**(4) database.

–e Causes **pg** not to pause at the end of each file.

–f Normally, **pg** splits lines longer than the screen width, but some sequences of characters in the text being displayed (such as escape sequences for underlining) generate undesirable results. The **–f** option inhibits **pg** from splitting lines.

–n Normally, commands must be terminated by a newline character. This option causes an automatic end of command as soon as a command letter is entered.

–s Causes **pg** to print all messages and prompts in standout mode (usually inverse video).

+linenumber
Start up at *linenumber*.

+/pattern/ Start up at the first line containing the regular expression pattern.

The responses you can type when **pg** pauses can be divided into three categories: those causing further perusal, those that search, and those that modify the perusal environment.

Commands that cause further perusal normally take a preceding *address*, which is an optionally signed number indicating the point from which further text should be displayed. This *address* is interpreted in either pages or lines depending on the command. A signed *address* specifies a point relative to the current page or line; an unsigned *address* specifies an address relative to the beginning of the file. Each command has a default address that is used if none is provided.

The perusal commands and their defaults are as follows:

(+1) newline or blank
: This causes one page to be displayed. The address is specified in pages.

(+1) **l** With a relative address, this causes **pg** to simulate scrolling the screen forward or backward the number of lines specified. With an absolute address, this command prints a screenful beginning at the specified line.

(+1) **d** or Ctrl-D
: Simulates scrolling half a screen forward or backward.

The following perusal commands take no address.

. or Ctrl-L
: Typing a single period causes the current page of text to be redisplayed.

$ Displays the last windowful in the file. Use with caution when the input is a pipe.

The following commands are available for searching for text patterns in the text. The regular expressions described in **ed**(1) are available. They must always be terminated by a newline, even if the **–n** option is specified.

i/*pattern*/
: Search forward for the *i*th (default *i*=1) occurrence of *pattern*. Searching begins immediately after the current page and continues to the end of the current file, without wrap-around.

i^*pattern*^
i?*pattern*?
: Search backwards for the *i*th (default *i*=1) occurrence of *pattern*. Searching begins immediately before the current page and continues to the beginning of the current file, without wrap-around. The ^ notation is useful for Adds 100 terminals which will not properly handle the ?.

After searching, **pg** will normally display the line found at the top of the screen. This can be modified by appending **m** or **b** to the search command to leave the line found in the middle or at the bottom of the window from now on. The suffix **t** can be used to restore the original situation.

The user of **pg** can modify the environment of perusal with the following commands:

*i***n** Begin perusing the *i*th next file in the command line. The *i* is an unsigned number; the default value is 1.

*i***p** Begin perusing the *i*th previous file in the command line. *i* is an unsigned number; the default is 1.

*i***w** Display another window of text. If *i* is present, set the window size to *i*.

s *filename*
Save the input in the named file. Only the current file being perused is saved. The white space between the **s** and *filename* is optional. This command must always be terminated by a newline, even if the **–n** option is specified.

h Help by displaying an abbreviated summary of available commands.

q or **Q** Quit **pg**.

!*command*
Command is passed to the shell, whose name is taken from the **SHELL** environment variable. If this is not available, the default shell is used. This command must always be terminated by a new-line, even if the **–n** option is specified.

At any time when output is being sent to the terminal, the user can hit the quit key (normally Ctrl-\) or the Interrupt (Break) key. This causes **pg** to stop sending output and display the prompt. The user can then enter one of the above commands in the normal manner. Unfortunately, some output is lost when this is done, due to the fact that any characters waiting in the terminal's output queue are flushed when the quit signal occurs.

If the standard output is not a terminal, then **pg** acts just like **cat**(1) except that a header is printed before each file (if there is more than one).

EXAMPLES

Following is a sample usage of **pg** in reading system news:

news | pg –p "(Page %d):"

FILES

/usr/lib/terminfo/?/*	Terminal information database
/tmp/pg*	Temporary file when input is from a pipe

SEE ALSO

ed(1), grep(1), terminfo(4)

NOTES

While waiting for terminal input, **pg** responds to Break, Delete, and ^ by terminating execution. Between prompts, however, these signals interrupt **pg**'s current task and place the user in prompt mode. These should be used with caution when input is being read from a pipe, since an interrupt is likely to terminate the other commands in the pipeline.

Users of Berkeley's **more** will find that the **z** and **f** commands are available, and that the terminal /, ^, or ? may be omitted from the searching commands.

BUGS

If terminal tabs are not set every eight positions, undesirable results may occur.

When using **pg** as a filter with another command that changes the terminal I/O options, terminal settings may not be restored correctly.

NAME

pmap – display or alter parameters of virtual-to-physical mapping devices

SYNOPSIS

/etc/pmap file [paddr size [type]]

DESCRIPTION

file must be a character special file associated with the **pmap** device driver (see **pmap**(4)). **pmap** sets the parameters of the indicated *file* to those specified (if any) and then displays all of the file's parameters.

The following parameters can be changed (only by the superuser, and only while the file is not in use):

paddr The starting physical address of the region associated with the file. Must be hexadecimal and must be a multiple of the system page size, as returned by **getpagesize**(2).

size The (hexadecimal) size of the region, in bytes. Must be hexadecimal and must be a multiple of the system page size.

type Must be either `npmem` (non-paged memory) or `phys` (physical). The PMAP_NPMEM bit for this file is set or cleared as indicated.

pmap also reports whether the file is currently mapped and whether it is currently open for exclusive access.

NOTE: This command allows the superuser to associate a file with any range of physical addesses, including MULTIBUS mapping registers, nonexistent memory, etc. Specifying incorrect values for *paddr* or *size* can lead to catastrophic results.

FILES

pmap files are application-specific

SEE ALSO

getpagesize(2), pmap(7)

DIAGNOSTICS

"Not a typewriter" indicates that the specified *file* is not a **pmap** device.

NAME

pr – print files

SYNOPSIS

pr [[**–column**] [**–w**width] [**–a**]] [**–e**ck] [**–i**ck] [**–drtfp**] [page] [**–n**ck] [**–o**offset] [**–l**length] [**–s**separator] [**–h**header] [file ...]

pr [[**–m**] [**–w**width]] [**–e**ck] [**–i**ck] [**–drtfp**] [page] [**–n**ck] [**–o**offset] [**–l**length] [**–s**separator] [**–h**header] file1 file2 ...

DESCRIPTION

pr is used to format and print the contents of a file. If *file* is – or if no files are specified, **pr** assumes standard input. **pr** prints the named files on standard output.

By default, the listing is separated into pages, each headed by the page number, the date and time the file was last modified, and the name of the file. Page length is 66 lines, which includes 10 lines of header and trailer output. The header is composed of 2 blank lines, 1 line of text (can be altered with **–h**), and 2 blank lines; the trailer is 5 blank lines. For single column output, line width may not be set and is unlimited. For multicolumn output, line width may be set and the default is 72 columns. Diagnostic reports (failed options) are reported at the end of standard output associated with a terminal, rather than interspersed in the output. Pages are separated by series of line feeds rather than form feed characters.

By default, columns are of equal width, separated by at least one space; lines which do not fit are truncated. If the **–s** option is used, lines are not truncated and columns are separated by the *separator* character.

Either *–column* or **–m** should be used to produce multi-column output. **–a** should be used only with *–column*, not with **–m**.

Command line options are as follows:

+page	Begin printing with page numbered *page* (default is 1).
–column	Print *column* columns of output (default is 1). Output appears as if **–e** and **–i** are turned on for multi-column output. May not use with **–m**.
–a	Print multi-column output across the page one line per column. *columns* must be greater than one. If a line is too long to fit in a column, it is truncated.
–m	Merge and print all files simultaneously, one per column. The maximum number of files that may be specified is eight. If a line is too long to fit in a column, it is truncated. May not use with *–column*.
–d	Double-space the output. Blank lines that result from double-spacing are dropped when they occur at the top of a page.
–e*ck*	Expand input tabs to character positions k+1, 2*k+1, 3*k+1, etc. If k is 0 or is omitted, default tab settings at every eighth position are assumed. Tab characters in the input are expanded into the appropriate number of spaces. If *c* (any non-digit character) is given, it is treated as the input tab character (default for *c* is

the tab character).

-i*ck* In output, replace white space wherever possible by inserting tabs to character positions *k*+1, 2**k*+1, 3**k*+1, and so on. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. If *c* (any non-digit character) is given, it is treated as the output tab character (default for *c* is the tab character).

-n*ck* Provide *k*-digit line numbering (default for *k* is 5). The number occupies the first *k*+1 character positions of each column of single column output or each line of **-m** output. If *c* (any non-digit character) is given, it is appended to the line number to separate it from whatever follows (default for *c* is a tab).

-w*width* Set the width of a line to *width* character positions (default is 72). This is effective only for multi-column output (*-column* and **-m**). There is no line limit for single column output.

-o*offset* Offset each line by *offset* character positions (default is 0). The number of character positions per line is the sum of the width and offset.

-l*length* Set the length of a page to *length* lines (default is 66). **-l**0 is reset to **-l**66. When the value of *length* is 10 or less, **-t** appears to be in effect since headers and trailers are suppressed. By default, output contains 5 lines of header and 5 lines of trailer leaving 56 lines for user-supplied text. When **-l***length* is used and *length* exceeds 10, then *length*-10 lines are left per page for user supplied text. When *length* is 10 or less, header and trailer output is omitted to make room for user supplied text.

-h *header* Use *header* as the text line of the header to be printed instead of the filename. **-h** is ignored when **-t** is specified or **-l***length* is specified and the value of *length* is 10 or less. (**-h** is the only **pr** option requiring space between the option and argument.)

-p Pause before beginning each page if the output is directed to a terminal (**pr** will ring the bell at the terminal and wait for a carriage return).

-f Use single form-feed character for new pages (default is to use a sequence of line-feeds). Pause before beginning the first page if the standard output is associated with a terminal.

-r Print no diagnostic reports on files that will not open.

-t Print neither the five-line identifying header nor the five-line trailer normally supplied for each page. Quit printing after the last line of each file without spacing to the end of the page. Use of **-t** overrides the **-h** option.

-s*separator* Separate columns by the single character *separator* instead of by the appropriate number of spaces (default for *separator* is a tab). Prevents truncation of lines on multicolumn output unless **-w** is specified.

EXAMPLES

Print *file1* and *file2* as a double-spaced, three-column listing headed by "file list":

pr –3dh "file list" file1 file2

Copy *file1* to *file2*, expanding tabs to columns 10, 19, 28, 37, . . . :

pr –e9 –t <file1 >file2

Print *file1* and *file2* simultaneously in a two-column listing with no header or trailer where both columns have line numbers:

pr –t –n file1 | pr –t –m –n file2 –

FILES

/dev/tty*	If standard output is directed to one of the special files */dev/tty**, then other output directed to this terminal is delayed until standard output is completed. This prevents error messages from being interspersed throughout the output.

SEE ALSO

cat(1), pg(1)

NAME

prof – display profile data

SYNOPSIS

prof [**–tcan**] [**–ox**] [**–g**] [**–z**] [**–h**] [**–s**] [**–m** mdata] [prog]

DESCRIPTION

The **prof** command interprets a profile file produced by the **monitor**(3C) function. The symbol table in the object file *prog* (*a.out* by default) is read and correlated with a profile file (*mon.out* by default). For each external text symbol, the percentage of time spent executing between the address of that symbol and the address of the next is printed, together with the number of times that function was called and the average number of milliseconds per call.

The mutually exclusive options **t, c, a,** and **n** determine the type of sorting of the output lines:

–t Sort by decreasing percentage of total time (default).

–c Sort by decreasing number of calls.

–a Sort by increasing symbol address.

–n Sort lexically by symbol name.

The mutually exclusive options **o** and **x** specify the printing of the address of each symbol monitored:

–o Print each symbol address (in octal) along with the symbol name.

–x Print each symbol address (in hexadecimal) along with the symbol name.

The following options may be used in any combination:

–g Include non-global symbols (static functions).

–z Include all symbols in the profile range [see **monitor**(3C)], even if associated with zero number of calls and zero time.

–h Suppress the heading normally printed on the report. (This is useful if the report is to be processed further.)

–s Print a summary of several of the monitoring parameters and statistics on the standard error output.

–m mdata Use file *mdata* instead of *mon.out* as the input profile file.

A program creates a profile file if it has been loaded with the **–p** option of **cc**(1). This option to the **cc** command arranges for calls to **monitor**(3C) at the beginning and end of execution. It is the call to **monitor** at the end of execution that causes a profile file to be written. The number of calls to a function is tallied if the **–p** option was used when the file containing the function was compiled.

The name of the file created by a profiled program is controlled by the environment variable **PROFDIR**. If this variable does not exist, *mon.out* is produced in the directory that is current when the program terminates. If **PROFDIR** = string, *string/pid.progname* is produced, where *progname* consists of argv[0] with any path prefix removed, and *pid* is the program's process id. If **PROFDIR** is the null string, no profiling output is produced.

A single function may be split into subfunctions for profiling by means of the MARK macro [see **prof**(5)]..

FILES

mon.out	For profile
a.out	For namelist

SEE ALSO

cc(1), exit(2), profil(2), monitor(3C), prof(5)

WARNINGS

The times reported in successive identical runs may show variances of 20% or more, because of varying cache-hit ratios due to sharing of the cache with other processes. Even if a program seems to be the only one using the machine, hidden background or asynchronous processes may blur the data. In rare cases, the clock ticks initiating recording of the program counter may "beat" with loops in a program, grossly distorting measurements.

Call counts are always recorded precisely.

The times for static functions are attributed to the preceding external text symbol if the **-g** option is not used. However, the call counts for the preceding function are still correct, (that is, the static function call counts are not added in with the call counts of the external function).

NOTES

Only programs that call **exit**(2) or return from **main** will cause a profile file to be produced, unless a final call to monitor is explicitly coded.

The use of the **-p** option to **cc**(1) to invoke profiling imposes a limit of 600 functions that may have call counters established during program execution. For more counters, you must call **monitor**(3C) directly. If this limit is exceeded, other data will be overwritten and the *mon.out* file will be corrupted. The number of call counters used will be reported automatically by the **prof** command whenever the number exceeds 5/6 of the maximum.

NAME

prompt – prompt program

SYNOPSIS

prompt [–T] *command* -- [*options*]

DESCRIPTION

The **prompt** program allows the user to initiate a command. The command's form is displayed on the terminal screen and the cursor is positioned in the first input field. User responses are edited according to the specifications in the command description file. Help is available at both the option and command level by pressing the HELP key. After all editing is complete, the command is executed.

–T The command is run in trace mode. This option tells the **prompt** command to display the text generated according to the DISPLAY TEMPLATE in the command description. The command is not executed. You can use trace mode in **vi** (the text editor) to format commands.

command Name of the command description file that the prompt command uses. This is a compiled file created by the **ec** command compiler.

options These options apply to the command called by the prompt command. Use a double hyphen to separate the options from the command name. The prompt command reads the command description file and then applies the options to the command. The prompt command lets the user select options before the form is displayed on the screen.

Environment Variables

TFTERM Identifies the terminal type.

TERM If **TFTERM** is not set, this variable identifies the terminal type.

TERMINFO Identifies an alternate location for the terminfo terminal descriptions. You do not need to define this environment variable since the directory *${TFADMFILE}/terminfo* is used by default.

TFADMFILE Identifies the directory where the menu system administrative data files are located. This is used to locate the subdirectory *edc*, the default location of compiled files created by the **ec** command.

TFEDCPATH Identifies directories that are alternate locations for the compiled files created by the **ec** command.

FILES

${TFADMFILE}/[a-z]*termtype*	Terminal definition file
${TFADMFILE}/edc/*command*	Compiled command description file

SEE ALSO

ec(1M), menu(1M)

NAME

prs – print an SCCS file

SYNOPSIS

prs [**–d**[dataspec]] [**–r**[SID]] [**–e**] [**–l**] [**–c**[date-time]] [**–a**] files

DESCRIPTION

prs prints, on the standard output, parts or all of an SCCS file [see **sccsfile**(4)] in a user-supplied format. If a directory is named, **prs** behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the pathname does not begin with *s.*), and unreadable files are silently ignored. If a name of – is given, the standard input is read. Each line of the standard input is taken to be the name of an SCCS file or directory to be processed; non-SCCS files and unreadable files are silently ignored.

Arguments to **prs**, which may appear in any order, consist of keyletter arguments, and filenames.

All the described keyletter arguments apply independently to each named file:

–d*[dataspec]* Used to specify the output data specification. The *dataspec* is a string consisting of SCCS file data keywords (see **Data Keywords**) interspersed with optional user supplied text.

–r*[SID]* Used to specify the SCCS identification (SID) string of a delta for which information is desired. If no SID is specified, the SID of the most recently created delta is assumed.

–e Requests information for all deltas created *earlier* than and including the delta designated via the **–r** keyletter or the date given by the **–c** option.

–l Requests information for all deltas created *later* than and including the delta designated via the **–r** keyletter or the date given by the **–c** option.

–c*[date-time]* The cutoff date-time **–c**[cutoff]] is in the form:

YY[MM[DD[HH[MM[SS]]]]]

Units omitted from the date-time default to their maximum possible values. For example, **–c7502** is equivalent to **–c750228235959**. Any number of non-numeric characters may separate the various 2-digit pieces of the *cutoff* date in the form **"–c77/2/2 9:22:25"**.

–a Requests printing of information for both removed (delta type = *R*) [see **rmdel**(1)] and existing (delta type = *D*) deltas. If the **–a** keyletter is not specified, information for existing deltas only is provided.

Data Keywords

Data keywords specify which parts of an SCCS file are to be retrieved and output. All parts of an SCCS file [see **sccsfile**(4)] have an associated data keyword. There is no limit on the number of times a data keyword may appear in a *dataspec*.

The information printed by **prs** consists of: (1) the user-supplied text; and (2) appropriate values (extracted from the SCCS file) substituted for the recognized data keywords in the order of appearance in the *dataspec*. The format of a data keyword value is either *Simple* (S), in which keyword substitution is direct, or *Multi-line* (M), in which keyword substitution is followed by a carriage return.

User-supplied text is any text other than recognized data keywords.

A tab is specified by **\t** and carriage return/newline is specified by **\n**. The default data keywords are as follows:

":Dt:\t:DL:\nMRs:\n:MR:COMMENTS:\n:C:"

SCCS Files Data Keywords

Keyword	Data Item	File Section	Value	Format
:Dt:	Delta information	Delta Table	See below*	S
:DL:	Delta line statistics	"	:Li:/:Ld:/:Lu:	S
:Li:	Lines inserted by Delta	"	nnnnn	S
:Ld:	Lines deleted by Delta	"	nnnnn	S
:Lu:	Lines unchanged by Delta	"	nnnnn	S
:DT:	Delta type	"	*D* or *R*	S
:I:	SCCS ID string (SID)	"	:R:.:L:.:B:.:S:	S
:R:	Release number	"	nnnn	S
:L:	Level number	"	nnnn	S
:B:	Branch number	"	nnnn	S
:S:	Sequence number	"	nnnn	S
:D:	Date Delta created	"	:Dy:/:Dm:/:Dd:	S
:Dy:	Year Delta created	"	nn	S
:Dm:	Month Delta created	"	nn	S
:Dd:	Day Delta created	"	nn	S
:T:	Time Delta created	"	:Th::Tm::Ts:	S
:Th:	Hour Delta created	"	nn	S
:Tm:	Minutes Delta created	"	nn	S
:Ts:	Seconds Delta created	"	nn	S
:P:	Programmer who created Delta	"	logname	S
:DS:	Delta sequence number	"	nnnn	S
:DP:	Predecessor Delta seq-no.	"	nnnn	S
:DI:	Sequence numbers of deltas included, excluded, ignored	"	:Dn:/:Dx:/:Dg:	S
:Dn:	Deltas included (seq #)	"	:DS: :DS:...	S
:Dx:	Deltas excluded (seq #)	"	:DS: :DS:...	S
:Dg:	Deltas ignored (seq #)	"	:DS: :DS:...	S
:MR:	MR numbers for delta	"	text	M

*** :Dt: = :DT: :I: :D: :T: :P: :DS: :DP:**

SCCS Files Data Keywords (Cont.)

Key-word	Data Item	File Section	Value	Format
:C:	Comments for delta	"	text	M
:UN:	User names	User Names	text	M
:FL:	Flag list	Flags	text	M
:Y:	Module type flag	"	text	S
:MF:	MR validation flag	"	*yes* or *no*	S
:MP:	MR validation pgm name	"	text	S
:KF:	Keyword error/warning flag	"	*yes* or *no*	S
:KV:	Keyword validation string	"	text	S
:BF:	Branch flag	"	*yes* or *no*	S
:J:	Joint edit flag	"	*yes* or *no*	S
:LK:	Locked releases	"	:R:...	S
:Q:	User-defined keyword	"	text	S
:M:	Module name	"	text	S
:FB:	Floor boundary	"	:R:	S
:CB:	Ceiling boundary	"	:R:	S
:Ds:	Default SID	"	:I:	S
:ND:	Null delta flag	"	*yes* or *no*	S
:FD:	File descriptive text	Comments	text	M
:BD:	Body	Body	text	M
:GB:	Gotten body	"	text	M
:W:	A form of *what*(1) string	N/A	:Z::M:\t:I:	S
:A:	A form of *what*(1) string	N/A	:Z::Y: :M: :I::Z:	S
:Z:	*what*(1) string delimiter	N/A	@(#)	S
:F:	SCCS filename	N/A	text	S
:PN:	SCCS file pathname	N/A	text	S

EXAMPLES

prs –d"Users and/or user IDs for :F: are:\n:UN:" s.file

may produce on the standard output:

```
Users and/or user IDs for s.file are:
xyz
131
abc
```

prs –d"Newest delta for pgm :M:: :I: Created :D: By :P:" –r s.file

may produce on the standard output:

```
Newest delta for pgm main.c: 3.7 Created 77/12/1 By cas
```

As a *special case,* **prs s.file** *may produce the following on the standard output for each delta table entry of the D type.*

```
D 1.1 77/12/1 00:00:00 cas 1 000000/00000/00000
MRs:
bl78-12345
bl79-54321
COMMENTS:
this is the comment line for s.file initial delta
```

The only keyletter argument allowed to be used with the *special case* is the **–a** keyletter.

FILES

/tmp/pr?????

SEE ALSO

admin(1), delta(1), get(1), help(1), sccsfile(4)

DIAGNOSTICS

Use **help**(1) for explanations.

NAME

prtvtoc – print the VTOC of a block device

SYNOPSIS

/etc/prtvtoc [**–s**] [**–h**] [**–T**] [**–t fstab**] [**–m mnttab**] device

DESCRIPTION

prtvtoc allows the contents of the VTOC (volume table of contents) to be viewed by a user for reference or verification. In addition to printing the contents of the VTOC, the command lists any free areas on the disk (that is, those without a partition).

The *device* on which the VTOC is installed may be specified as a simple disk name such as **zd0** or **sd1** without any path prefix. If this form of *disk* is used, the appropriate path to a special device is generated automatically. Alternatively, the pathname to a character special device should be given. The character special device may be a disk partition, the disk's diagnostic device, or the prepartitioned disk device. (See **vtoc**(7)).

The following options to **prtvtoc** can be used:

–s Print only the partition table, with headings but without the comments. This option also inhibits the printing of all V_NOPART partitions.

–h Print only the partition table, without headings and comments. Use this option when the output of the **prtvtoc** command is piped into another command.

–T Print only the disk type name. Use this option to discover what the disk type is.

–t*fstab* Use the file *fstab* instead of */etc/fstab*.

–m*mnttab* Use the file *mnttab* instead of */etc/mnttab*.

EXAMPLE

The command line entry and system response shown below are for a 792-megabyte disk:

```
# prtvtoc /dev/rdsk/zd1s0 <CR>
* /dev/rdsk/zd1s0 partition map
*
* Disk Type:            m2333k
*
* Dimensions:
*     512 bytes/sector
*      66 sectors/track
*      10 tracks/cylinder
*     823 cylinders
*
* Partition Types:
*  0: Empty Slot
*  1: Regular Partition
*  2: Bootstrap Area
*  3: Reserved Area
*  4: Firmware Area
```

*	Type	Start Sector	Size in Sectors	Block Sz in Bytes	Frag Sz in Bytes	Mount Point
0	1	221100	49500	8192	1024	
1	1	270600	67320	8192	1024	
2	1	337920	203940	8192	1024	
3	1	692	220408	8192	1024	
4	0	0	0	8192	1024	
5	0	0	0	8192	1024	
6	0	0	0	8192	1024	
7	0	0	0	8192	1024	
8	0	0	0	8192	1024	
9	0	0	0	8192	1024	
10	0	0	0	8192	1024	
11	0	0	0	8192	1024	
12	3	0	660	0	0	
13	3	541860	1320	0	0	
14	2	660	16	0	0	
15	3	676	16	0	0	

#

Codes for TYPE are as follows:

Name	Number
No Partition	0
Regular Partition	1
Boot Area	2
Reserved Area	3
Firmware Area	4

If the *device* passed to **prtvtoc** does not contain a VTOC, an error will occur.

SEE ALSO

mkvtoc(1M)

NOTES

If a diagnostic device is provided to **prtvtoc** (i.e., a device of the form */dev/diag/rdsk/xx?)*, then mountpoints will not be listed.

Only users belonging to group *sys* can use this command.

NAME

ps – report process status

SYNOPSIS

ps [options]

DESCRIPTION

ps prints certain information about active processes. Without *options*, information is printed about processes associated with the controlling terminal. The output consists of a short listing containing only the process ID, terminal identifier, cumulative execution time, and the command name. Otherwise, the information that is displayed is controlled by the selection of *options*.

Options accept names or lists as arguments. Arguments can be either separated from one another by commas or enclosed in double quotes and separated from one another by commas or spaces. Values for *proclist* and *grplist* must be numeric.

The *options* are given in descending order according to volume and range of information provided:

–e Print information about every process now running.

–d Print information about all processes except process group leaders.

–a Print information about all processes most frequently requested: all those except process group leaders and processes not associated with a terminal.

–f Generate a full listing. (See below for significance of columns in a full listing.)

–F Generate an even fuller listing. (See below for significance of columns in a fuller listing.)

–l Generate a long listing. (See below.)

–n *name* Valid only for users with a real user ID of *root* or a real group ID of *sys*. Takes argument signifying an alternate system *name* in place of */unix*.

–t *termlist* List only process data associated with the terminal given in *termlist*. Terminal identifiers may be specified in one of two forms: the device filename (such as *ttyGA/GAAB*) or, if the device filename starts with *tty*, just the digit identifier (such as *GA/GAAB*).

–p *proclist* List only process data whose process ID numbers are given in *proclist*.

–u *uidlist* List only process data whose user ID number or login name is given in *uidlist*. In the listing, the numerical user ID will be printed unless you give the **–f** or **–F** option, which prints the login name.

–g *grplist* List only process data whose process group leader ID number(s) appears in *grplist*. (A group leader is a process whose process ID number is identical to its process group ID number. A login shell is a common example of a process group leader.)

Under the **-f** or **-F** options, **ps** tries to determine the command name and arguments given when the process was created by examining the user block. Failing this, the command name is printed as it would have appeared without the **-f** or **-F** options, in square brackets.

The column headings and the meaning of the columns in a **ps** listing are given below. The letters **f, F** and **l** indicate the option (**full, fuller** or **long**, respectively) that causes the corresponding heading to appear; **all** means that the heading always appears. Note that these two options determine only what information is provided for a process. They do not determine which processes will be listed.

F (l) Flags (hexadecimal and additive) associated with the process:

- 00 Process has terminated; process table entry now available.
- 01 A system process; always in primary memory.
- 02 Parent is tracing process.
- 04 Tracing parent's signal has stopped process; parent is waiting [**ptrace**(2)].
- 08 Process is currently in primary memory.
- 10 Process currently in primary memory; locked until an event completes.

S (l) The state of the process:

- O Process is running on a processor.
- S Sleeping; process is waiting for an event to complete.
- R Runable; process is on run queue.
- I Idle; process is being created.
- Z Zombie state; process terminated and parent not waiting.
- T Traced; process stopped by a signal because parent is tracing it.
- X SXBRK state; process is waiting for more primary memory.

UID (f,l) The user ID number of the process owner (the login name is printed under the **-f** option).

PID (all) The process ID of the process (this datum is necessary in order to kill a process).

PPID (f,l) The process ID of the parent process.

C (f,l) Processor utilization for scheduling.

PRI (l) The priority of the process (higher numbers mean lower priority).

NI (l) Nice value, used in priority computation.

ADDR (l) The memory address of the process.

SZ (l) The size (in K) of the swappable process's image in main memory.

RSS (F) The real memory (resident set) size of the process (in K) or, if the process is swapped, the resident set size when the

process was swapped out.

ENG (F) The processor number on which a runnable process is running (blank if the process is not running).

WCHAN (l) The address of an event for which the process is sleeping or in SXBRK state (if blank, the process is running).

STIME (f) The starting time of the process, given in hours, minutes, and seconds. (A process begun more than 24 hours before the **ps** inquiry is executed is given in months and days.)

TTY (all) The controlling terminal for the process (the message **?** prints when there is no controlling terminal).

TIME (all) The cumulative execution time for the process.

COMMAND (all) The command name (the full command name and its arguments are printed under the **–f** option).

A process that has exited and has a parent, but has not yet been waited for by the parent, is marked `<defunct>`.

FILES

/dev	
/dev/tty*	
/dev/kmem	Kernel virtual memory
/dev/swap	The default swap device
/dev/mem	Memory
/etc/passwd	UID information supplier
/etc/ps_data	Internal data structure
/unix	System namelist

SEE ALSO

kill(1), nice(1), getty(1M)

WARNINGS

Things can change while **ps** is running; the snap-shot it gives is only true for a split-second, and it may not be accurate by the time you see it. Some data printed for defunct processes is irrelevant.

If no *termlist*, *proclist*, *uidlist*, or *grplist* is specified, **ps** checks *stdin*, *stdout*, and *stderr* in that order, looking for the controlling terminal, and attempts to report on processes associated with the controlling terminal. In this situation, if *stdin*, *stdout*, and *stderr* are all redirected, **ps** will not find a controlling terminal, so there will be no report.

On a heavily loaded system, **ps** may report an **lseek**(2) error and exit. **ps** may seek to an invalid user area address: having obtained the address of a process' user area, **ps** may not be able to seek to that address before the process exits and the address becomes invalid.

ps –ef may not report the actual start of a tty login session, but rather an earlier time, when a getty was last respawned on the tty line.

If the user specifies the **–n** flag, the real and effective UID/GID will be set to the real UID/GID of the user invoking **ps**.

NAME

pwck, grpck – password/group file checkers

SYNOPSIS

/etc/pwck [file]
/etc/grpck [file]

DESCRIPTION

pwck scans the password file and notes any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and the program-to-use-as-Shell exist. The default password file is */etc/passwd*. The */etc/passwd* file contains the following entries that have names longer than 8 characters: checkfsys, mountsys, powerdown, and umountfsys. Only the first 8 characters are significant, therefore **pwck** generates warning messages because these names are longer than 8 characters. You can ignore these messages.

grpck verifies all entries in the group file. This verification includes a check of the number of fields, group name, group ID, and whether all login names appear in the password file. The default group file is */etc/group*. If a group contains no users, the error message `Null login name` is generated. The operating system is distributed with the group **other** which has no users and causes this message to be generated. You can ignore the message.

FILES

/etc/group
/etc/passwd

SEE ALSO

group(4), passwd(4)

DIAGNOSTICS

Group entries in */etc/group* with no login names are flagged.

NOTES

The standard password file, */etc/passwd,* contains the following entries that have names longer than 8 characters: checkfsys, mountfsys, powerdown, and umountfsys. Only the first 8 characters are significant in any name. **pwck** generates warnings because these names exceed 8 characters. You can ignore the warnings.

By default, the operating system includes a group called *other* containing no users. If a group contains no users, **grpck** generates the warning message `Null login name`. You can ignore the message.

NAME

pwconv – installs and updates */etc/shadow* with information from */etc/passwd*

SYNOPSIS

pwconv

DESCRIPTION

The **pwconv** command creates and updates */etc/shadow* with information from */etc/passwd*.

If the */etc/shadow* file does not exist, this command will create */etc/shadow* with information from */etc/passwd*. The command populates */etc/shadow* with the user's login name, password, and password aging information. If password aging information does not exist in */etc/passwd* for a given user, none will be added to */etc/shadow*. However, the "last changed" information will always be updated.

If the */etc/shadow* file does exist, the following tasks will be performed:

Entries that are in the */etc/passwd* file and not in the */etc/shadow* file will be added to the */etc/shadow* file.

Entries that are in the */etc/shadow* file and not in the */etc/passwd* file will be removed from */etc/shadow*.

Password attributes (for example, password and aging information) that exist in an */etc/passwd* entry will be moved to the corresponding entry in */etc/shadow*.

The **pwconv** program is a privileged system command that cannot be executed by ordinary users.

FILES

/etc/passwd, /etc/shadow, /etc/opasswd, /etc/oshadow

SEE ALSO

passwd(1), passmgmt(1M), pwunconv(1M)

DIAGNOSTICS

The **pwconv** command exits with one of the following values:

0	Success.
1	Permission denied.
2	Invalid command syntax.
3	Unexpected failure. Conversion not done.
4	Unexpected failure. Password file(s) missing.
5	Password file(s) busy. Try again later.

NAME

pwd – working directory name

SYNOPSIS

pwd

DESCRIPTION

pwd prints the pathname of the working (current) directory.

SEE ALSO

cd(1)

DIAGNOSTICS

`Cannot open ..` and `Read error in ..` indicate possible filesystem trouble and should be referred to a system administrator.

NAME

pwdhd – daemon to oversee password database creation

SYNOPSIS

pwdhd

DESCRIPTION

The **pwdhd** program is intended to be run as a daemon that is started by the **init** process. This is accomplished by an entry in the */etc/inittab* file.

The program will create the FIFO (named pipe) */etc/pwdp* if it does not exist. The program then checks for the existence of the password databases. If they do not exist, the program **/etc/mkpwdbm** is run as a child process to create them.

Next, the program reads from the FIFO looking for the characters **r**, **p**, and **s**. These characters cause **/etc/mkpwdbm** to rebuild the password database files. A **p** specifies the database file for */etc/passwd* is to be built, an **s** specifies the database file for */etc/shadow* is to be built, and an **r** indicates that both files are to be built.

The **pwdhd** program is intended to be invoked only by **init**. It must execute as the superuser to be able to run the program **/etc/mkpwdbm**.

FILES

/etc/mkpwdbm
/etc/pwdp
/etc/passwd
/etc/passwd_dbm.dir
/etc/passwd_dbm.pag
/etc/shadow
/etc/shadow_dbm.dir
/etc/shadow_dbm.pag

SEE ALSO

/etc/mkpwdbm

NAME

pwfix – remove incorrectly formatted password entries

SYNOPSIS

pwfix [-fqv]

DESCRIPTON

The **pwfix** command scans the password file */etc/passwd,* and optionally */etc/shadow,* allowing any incorrectly formatted entries to be deleted. **pwfix** should be run when an error is reported by one of the programs which manipulates the password files.

pwfix uses **fgetpwent**(3C) and **fgetspent**(3C) to read the password file (and shadow file) entries one at a time and copy them to a temporary file. If these routines cannot parse an entry, the entry is an error and must be deleted or corrected. After completing a scan of the file, the temporary file is renamed to the original.

By default the program runs interactively. If a bad entry is encountered, it is displayed and the user prompted with a question asking if the line should be deleted. A response of **y** deletes the entry, **n** keeps the line, and **q** terminates the scan of the file. A count of the deleted lines is provided. If no lines are deleted, there is no output.

The **-f** option forces deletion of bad lines. If **-f** is specified, **pwfix** lists the lines which were deleted and a count of the lines deleted. If no lines are deleted, there is no output.

The **-q** option is used to suppress summary information. It is normally combined with **-f** when the command is used in a shell script.

The **-v** option is used to force summary information.

This command may be used only by the super user.

This program allows recovery of damaged password files. Deleted lines can be replaced by using system administration tools. Hand editing must be done only with great care and is not recommended.

FILES

/etc/passwd
/etc/shadow
/etc/opasswd
/etc/oshadow

SEE ALSO

passmgmt(1M), passwd(1M), pwconv(1M), pwunconv(1M), passwd(4)

DIAGNOSTICS

The pwfix command exits with one of the following values:

0 - Success, no files changed
1 - No permission
2 - Lock file busy
3 - Incorrect command syntax
4 - File error occurred
5 - Success, files were changed

NAME

pwunconv – converts from a two to a one password file scheme

SYNOPSIS

pwunconv

DESCRIPTION

pwunconv converts the operating system from a two password file scheme (*/etc/passwd* and */etc/shadow*) to a one password file scheme (*/etc/passwd*). It updates */etc/passwd* with password information from */etc/shadow*. If aging information is present in */etc/shadow*, the password aging information in */etc/passwd* is also updated.

The total size of a login entry for the password file is limited to a maximum of 511 bytes.

FILES

/etc/passwd, /etc/shadow, /etc/opasswd, /etc/oshadow

SEE ALSO

passwd(1M), passmgmt(1M), pwconv(1M)

DIAGNOSTICS

The **pwunconv** command exits with one of the following values:

0	Success.
1	Permission denied.
2	Invalid command syntax.
3	Unexpected failure. Conversion not done.
4	Unexpected failure. Password file missing.
5	Password file(s) busy. Try again later.
6	Shadow password file does not exist.
7	Entry for *name* too long. Conversion not done.

NAME

quota – display disk usage and limits

SYNOPSIS

quota [**–v**] [*user*]

DESCRIPTION

The **quota** command displays users' disk usage and limits. Only the superuser can use the optional *user* argument to view the limits of other users.

When the **quota** command is issued without options, it displays warnings only about mounted filesystems where usage is over quota. Remotely mounted filesystems that are mounted with the **noquota** option (see **fstab**(4)) are ignored.

The following option is available:

–v Display user's quotas on all mounted filesystems where quotas exist

FILES

quotas	Quota file at the filesystem root
/etc/mnttab	List of currently mounted filesystems

SEE ALSO

edquota(1M), quotaon(1M), quotactl(2seq), fstab(4)

NAME

quotacheck – filesystem quota consistency checker

SYNOPSIS

/usr/etc/quotacheck [**–v**] [**–p**] filesystem...
/usr/etc/quotacheck [**–v**] [**–p**] **–a**

DESCRIPTION

quotacheck examines each filesystem, builds a table of current disk usage, and compares this table against that stored in the disk quota file for the filesystem. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated. (The latter occurs only if an active filesystem is checked).

quotacheck expects each filesystem to be checked to have a quota file named *quotas* in the root directory. If none is present, **quotacheck** ignores the filesystem.

quotacheck is normally run at boot time from the */etc/rc2.d/S01MOUNTFSYS* file (see **rc2**(1M)) before disk quotas are enabled with **quotaon**(1M).

quotacheck accesses the raw device in calculating the actual disk usage for each user. Thus, the filesystems checked should be quiescent while **quotacheck** is running.

The following options are available:

–v Indicate the calculated disk quotas for each user on a particular filesystem. **quotacheck** normally reports only those quotas modified.

–a Check all the filesystems whose entries in */etc/fstab* indicate they allow read-write access and have disk quotas.

–p Run parallel passes on the required filesystems, using the pass numbers in */etc/fstab* in the same way they are used with **fsck**(1M).

FILES

/etc/fstab Default filesystems

SEE ALSO

quotaon(1M), quotactl(2seq)

NAME

quotaon, quotaoff – turn filesystem quotas on and off

SYNOPSIS

/usr/etc/quotaon [**–v**] *filesystem...*

/usr/etc/quotaon [**–v**] **–a**

/usr/etc/quotaoff [**–v**] *filesystem...*

/usr/etc/quotaoff [**–v**] **–a**

DESCRIPTION

quotaon announces to the system that disk quotas should be enabled on one or more filesystems. The filesystems specified must have entries in */etc/fstab* and must be mounted at the time. The filesystem quota files must be present in the root directory of the specified filesystem and must be named *quotas*. The optional **–v** argument causes **quotaon** to print a message for each filesystem that has quotas turned on. If **quotaon** has an **–a** argument instead of a list of filesystems, quotas are turned on for all filesystems whose entries in */etc/fstab* indicate that they allow read-write access and have quotas. This option is normally used to enable quotas at boot time.

quotaoff announces to the system that the specified filesystems should have any disk quotas turned off. As above, the **–v** option forces a verbose message for each filesystem affected, and the **–a** option disables quotas for all filesystems listed in */etc/fstab*.

These commands update the status field of devices located in */etc/mnttab* to indicate when quotas are on or off for each filesystem.

FILES

/etc/mnttab	Mount table
/etc/fstab	Filesystem table

SEE ALSO

edquota(1M), quotactl(2SEQ), fstab(4), mnttab(4)

NAME

ratfor – rational Fortran dialect

SYNOPSIS

ratfor [options] [files]

DESCRIPTION

ratfor converts a dialect of FORTRAN into ordinary FORTRAN. **ratfor** provides control flow constructs essentially identical to those in C:

statement grouping:
{ statement; statement; statement }

decision-making:
if (condition) statement [**else** statement]
switch (integer value) {
case integer: statement
...
[**default**:] statement
}

loops:
while (condition) statement
for (expression; condition; expression) statement
do limits statement
repeat statement [**until** (condition)]
break
next

and some syntactic sugar to make programs easier to read and write:

free form input:
multiple statements/line; automatic continuation

comments:
this is a comment.

translation of relationals:
>, >=, etc., become .GT., .GE., etc.

return expression to caller from function:
return (expression)

define:
define *name replacement*

include:
include *file*

The option **–h** causes quoted strings to be turned into **27H** constructs. The **–C** option copies comments to the output and attempts to format it neatly. Normally, continuation lines are marked with a & in column 1; the option **–6x** makes the continuation character **x** and places it in column 6.

ratfor is best used with **f77**(1).

SEE ALSO

efl(1), f77(1)

NAME

rc0 – run commands performed to stop the operating system

SYNOPSIS

/etc/rc0

DESCRIPTION

This file is executed at each system state change that needs to have the system in an inactive state. It is responsible for those actions that bring the system to a state where it is safe to remove power, traditionally called *shutdown*.

There are three system states that require this procedure. They are state 0 (the system halt state), state 5 (the firmware state), and state 6 (the reboot state). Whenever a change to one of these states occurs, the **/etc/rc0** procedure is run. The entry in */etc/inittab* might read as follows:

```
s0:056:wait:/etc/rc0 >/dev/console 2>&1 </dev/console
```

Some of the actions performed by **/etc/rc0** are carried out by files in the directory */etc/shutdown.d* and files beginning with *K* in */etc/rc0.d*. These files are executed in ASCII order (see **FILES** below for more information), terminating some system service. The combination of commands in **/etc/rc0** and files in */etc/shutdown.d* and */etc/rc0.d* determine how the system is shut down.

The recommended sequence for **/etc/rc0** is as follows:

Stop System Services and Daemons.

Various system services (such as the LP Spooler) are gracefully terminated.

When new services are added that should be terminated when the system is shut down, the appropriate files are installed in */etc/shutdown.d* and */etc/rc0.d*.

Terminate Processes

SIGTERM signals are sent to all running processes by **killall**(1M). Processes stop themselves cleanly if sent SIGTERM.

Kill Processes

SIGKILL signals are sent to all remaining processes; no process can resist SIGKILL.

At this point the only processes left are those associated with **/etc/rc0** and processes 0 and 1, which are special to the operating system.

Unmount All Filesystems

Only the root filesystem (*/*) remains mounted.

Depending on which system state the systems end up in (0, 5, or 6), the entries in */etc/inittab* direct what happens next. If */etc/inittab* has not defined any other actions to be performed, as in the case of system state 0, the operating system will have nothing to do. It should not be possible to get the system's attention. The only thing that can be done is to turn off the power or possibly get the attention of a firmware monitor. The **shutdown** command can be used only by the superuser.

FILES

The execution by **/bin/sh** of any files in */etc/shutdown.d* occurs in ascii sort-sequence order. See **rc2**(1M) for more information.

SEE ALSO

killall(1M), rc2(1M), shutdown(1M), rc(4)

NAME

rc2 – run commands performed for multi-user environment

SYNOPSIS

/etc/rc2

DESCRIPTION

This file is executed by an entry in */etc/inittab* and initializes the system to a ready-to-use state, traditionally state 2, called the *multiuser* state.

The actions performed by **/etc/rc2** are found in files in the directory */etc/rc.d* and files beginning with *S* in */etc/rc2.d*. These files are executed by **/bin/sh** in ASCII sort–sequence order (see **FILES** for more information).

You can create custom files (shell scripts) for your system in the directory */etc/rc2.d*. If, during initialization to multi-user state, any problem is encountered that would make it detrimental to enter multiuser state, the custom file could create a file called *.single_user* in the directory */etc/rc2.d*. The file can contain text to explain the failure in more detail, and its contents can be displayed when the system is returned to single-user mode.

For example, this mechanism is currently used by MOUNTFSYS in */etc/rc2*. If problems are encountered during filesystem mounting, the message `Going back to SINGLE USER mode!` is issued, the contents of *.single_user* are displayed, and the system is returned to single-user mode. The following code is part of the MOUNTFSYS routine:

```
# If the .single_user file gets created, it will contain
# the reason to avoid going multi-user.  Send init the
# SINGLE_USER signal, print out the reason for going back
# to single user mode and exit.  Init cannot proceed with
# the run-level change until this script exits.

/bin/rm -f /etc/rc2.d/.single_user
if [ -d /etc/rc2.d ]
then
      for f in /etc/rc2.d/S*
      {
            if [ -s ${f} ]
            then
                  /bin/sh ${f} start
                  status=$?
                  if [ -f /etc/rc2.d/.single_user ]
                  then
                        /etc/init S
                        echo '*** Going back to SINGLE
                                    USER mode!'
                        /bin/cat /etc/rc2.d/.single_user
                        exit $status
                  fi
            fi
      }
fi
```

The functions done by the **/etc/rc2** command and associated */etc/rc2.d* files include the following:

Setting and exporting the TIMEZONE variable.

Setting-up and mounting the user (*/usr*) filesystem.

Cleaning up the */tmp* and */usr/tmp* directories.

Starting the **cron** daemon by executing **/etc/cron**.

Cleaning up (deleting) uucp locks status and temporary files in the */usr/spool/uucp* directory.

Other functions can be added, as required, to support the addition of hardware and software features.

EXAMPLES

The following are prototypical files found in */etc/rc2.d*. These files are prefixed by an *S* and a number indicating the execution order of the files.

RMTMPFILES

```
#  clean up /tmp
rm -rf /tmp
mkdir /tmp
chmod 777 /tmp
chgrp sys /tmp
chown sys /tmp
```

uucp

```
#    clean-up uucp locks, status, and temporary files
rm -rf /usr/spool/locks/*
```

The file */etc/TIMEZONE* is included early in */etc/rc2*, thus establishing the default time zone for all commands that follow.

FILES

Here are some hints about files in */etc/rc.d*:

The order in which files are executed is important. Since they are executed in ASCII sort–sequence order, using the first character of the filename as a sequence indicator helps keep the proper order. Thus, files starting with the following characters would be:

[0-9]. very early
[A-Z]. early
[a-n]. later
[o-z]. last

Files in */etc/rc.d* that begin with a dot (.) are not executed. This feature can be used to hide files that are not to be executed for the time being without removing them. The command can be used only by the superuser.

Files in */etc/rc2.d* must begin with an *S* or a *K* followed by a two-digit sequence number and the rest of the filename. Upon entering run level 2, files beginning with *S* are executed with the **start** option; files beginning with *K* are executed with the **stop** option. Files beginning with other characters are ignored.

SEE ALSO

rc0(1M), shutdown(1M), rc(4)

NAME

rcs – change RCS file attributes

SYNOPSIS

rcs [options] file ...

DESCRIPTION

rcs creates new RCS files or changes attributes of existing ones. An RCS file contains multiple revisions of text, an access list, a change log, descriptive text, and some control attributes. For **rcs** to work, the caller's login name must be on the access list; if the access list is empty, the caller must be the owner of the file or the superuser, or the **–i** option must be present.

Files ending in *,v* are RCS files, all others are working files. If a working file is given, **rcs** tries to find the corresponding RCS file first in directory *./RCS* and then in the current directory, as explained in **co**(1).

–i Creates and initializes a new RCS file, but does not deposit any revision. If the RCS file has no path prefix, **rcs** tries to place it first into the subdirectory *./RCS*, and then into the current directory. If the RCS file already exists, an error message is printed.

–a*logins* Appends the login names appearing in the comma-separated list *logins* to the access list of the RCS file.

–A*oldfile* Appends the access list of *oldfile* to the access list of the RCS file.

–e[*logins*] Erases the login names appearing in the comma-separated list *logins* from the access list of the RCS file. If *logins* is omitted, the entire access list is erased.

–c*string* Sets the comment leader to *string*. The comment leader is printed before every log message line generated by the keyword Log during checkout (see **co**). This is useful for programming languages without multi-line comments. During **rcs -i** or initial **ci**, the comment leader is guessed from the suffix of the working file.

–l[*rev*] Locks the revision with number *rev*. If a branch is given, the latest revision on that branch is locked. If *rev* is omitted, the latest revision on the trunk is locked. Locking prevents overlapping changes. A lock is removed with **ci** or **rcs –u** (see below).

–u[*rev*] Unlocks the revision with number *rev*. If a branch is given, the latest revision on that branch is unlocked. If *rev* is omitted, the latest lock held by the caller is removed. Normally, only the locker of a revision may unlock it. Somebody else unlocking a revision breaks the lock. This causes a mail message to be sent to the original locker. The message contains a commentary solicited from the breaker. The commentary is terminated with a line containing a single dot (.) or Ctrl-D.

–L Sets locking to *strict*. Strict locking means that the owner of an RCS file is not exempt from locking for check-in. This option

should be used for files that are shared.

–U Sets locking to non-strict. Non-strict locking means that the owner of a file need not lock a revision for checkin. This option should NOT be used for files that are shared. The default (**–L** or **–U**) is determined by your system administrator.

–n*name*[*:rev*] Associates the symbolic name *name* with the branch or revision *rev*. **rcs** prints an error message if *name* is already associated with another number. If *rev* is omitted, the symbolic name is deleted.

–N*name*[*:rev*] Same as **–n**, except that it overrides a previous assignment of *name*.

–o*range* Deletes (outdates) the revisions given by *range*. A range consisting of a single revision number means that revision. A range consisting of a branch number means the latest revision on that branch. A range of the form *rev1–rev2* means revisions *rev1* to *rev2* on the same branch, *–rev* means from the beginning of the branch containing *rev* up to and including *rev*, and *rev–* means from revision *rev* to the end of the branch containing *rev*. Outdated revisions may not have branches or locks.

–q Quiet mode; diagnostics are not printed.

–s*state*[*:rev*] Sets the state attribute of the revision *rev* to *state*. If *rev* is omitted, the latest revision on the trunk is assumed. If *rev* is a branch number, the latest revision on that branch is assumed. Any identifier is acceptable for *state*. A useful set of states is *Exp* (for experimental), *Stab* (for stable), and *Rel* (for released). By default, **ci** sets the state of a revision to *Exp*.

–t[*txtfile*] Writes descriptive text into the RCS file (deletes the existing text). If *txtfile* is omitted, **rcs** prompts the user for text supplied from the standard input, terminated with a line containing a single '.' or Ctrl-D. Otherwise, the descriptive text is copied from the file *txtfile*. If the **–i** option is present, descriptive text is requested even if **–t** is not given. The prompt is suppressed if the standard input is not a terminal.

FILES

The caller of the command must have read/write permission for the directory containing the RCS file and read permission for the RCS file itself. **rcs** creates a semaphore file in the same directory as the RCS file to prevent simultaneous update. For changes, **rcs** always creates a new file. On successful completion, **rcs** deletes the old one and renames the new one. This strategy makes links to RCS files useless.

SEE ALSO

co(1), ci(1), ident(1), rcsdiff(1), rcsintro(1), rcsmerge(1), rlog(1), sccstorcs(1), rcsfile(4)

DIAGNOSTICS

The RCS filename and the revisions outdated are written to the diagnostic output. The exit status always refers to the last RCS file operated on and is 0 if the operation was successful, 1 otherwise.

NAME

rcsdiff – compare RCS revisions

SYNOPSIS

rcsdiff [**-biwt**] [**-cefhn**] [**-r***rev1*] [**-r***rev2*] **file ...**

DESCRIPTION

rcsdiff runs **diff**(1) to compare two revisions of each RCS file given. A filename ending in *,v* is an RCS filename, otherwise a working filename. **rcsdiff** derives the working filename from the RCS filename and vice versa, as explained in **co**(1). Pairs consisting of both an RCS and a working filename may also be specified.

All options except **-r** have the same effect as described in **diff**(1).

If both *rev1* and *rev2* are omitted, **rcsdiff** compares the latest revision on the trunk with the contents of the corresponding working file. This is useful for determining what you changed since the last check-in.

If *rev1* is given but *rev2* is omitted, **rcsdiff** compares revision *rev1* of the RCS file with the contents of the corresponding working file.

If both *rev1* and *rev2* are given, **rcsdiff** compares revisions *rev1* and *rev2* of the RCS file.

Both *rev1* and *rev2* may be given numerically or symbolically.

EXAMPLES

The following command runs **diff** on the latest trunk revision of RCS file *f.c,v* and the contents of working file *f.c*.

```
rcsdiff f.c
```

SEE ALSO

ci(1), co(1), diff(1), ident(1), rcs(1), rcsintro(1), rcsmerge(1), rlog(1), rcsfile(4)

NOTES

Copyright 1982 by Walter F. Tichy.

NAME

rcsintro - introduction to RCS commands

DESCRIPTION

The Revision Control System (RCS) manages multiple revisions of text files. RCS automates the storing, retrieval, logging, identification, and merging of revisions. RCS is useful for text that is revised frequently, for example programs, documentation, graphics, papers, form letters, and so on.

The basic user interface is extremely simple. The novice needs to learn only two commands: **ci**(1) and **co**(1). **ci**, short for "check in," deposits the contents of a text file into an archival file called an RCS file. An RCS file contains all revisions of a particular text file. **co**, short for "check out," retrieves revisions from an RCS file.

Functions of RCS

- Storage and retrieval of multiple revisions of text. RCS saves all old revisions in a space efficient way. Changes no longer destroy the original, because the previous revisions remain accessible. Revisions can be retrieved according to ranges of revision numbers, symbolic names, dates, authors, and states.
- Maintenance of a complete history of changes. RCS logs all changes automatically. Besides the text of each revision, RCS stores the author, the date and time of check-in, and a log message summarizing the change. The logging makes it easy to find out what happened to a module, without having to compare source listings or having to track down colleagues.
- Resolution of access conflicts. When two or more programmers wish to modify the same revision, RCS alerts the programmers and prevents one modification from corrupting the other.
- Maintenance of a tree of Revisions. RCS can maintain separate lines of development for each module. It stores a tree structure that represents the ancestral relationships among revisions.
- Merging of revisions and resolution of conflicts. Two separate lines of development of a module can be coalesced by merging. If the revisions to be merged affect the same sections of code, RCS alerts the user about the overlapping changes.
- Release and configuration control. Revisions can be assigned symbolic names and marked as released, stable, experimental, and so on. With these facilities, configurations of modules can be described simply and directly.
- Automatic identification of each revision with name, revision number, creation time, author, etc. The identification is like a stamp that can be embedded at an appropriate place in the text of a revision. The identification makes it simple to determine which revisions of which modules make up a given configuration.
- Minimization of secondary storage. RCS needs little extra space for the revisions (only the differences). If intermediate revisions are deleted, the corresponding deltas are compressed accordingly.

Getting Started with RCS

Suppose you have a file *f.c* that you wish to put under control of RCS. Invoke the check-in command:

ci f.c

This command creates the RCS file *f.c,v*, stores *f.c* in it as revision 1.1, and deletes *f.c*. It also asks you for a description. The description should be a synopsis of the contents of the file. All later check-in commands will ask you for a log entry, which should summarize the changes that you made.

Files ending in *,v* are called RCS files (*v* stands for *versions*), the others are called working files. To get back the working file *f.c* in the previous example, use the check-out command:

co f.c

This command extracts the latest revision from *f.c,v* and writes it into *f.c*. You can now edit *f.c* and check it back in by invoking this command:

ci f.c

ci increments the revision number properly. If **ci** complains with the following message, then your system administrator has decided to create all RCS files with the locking attribute set to *strict*.

```
ci error: no lock set by <your login>
```

In this case, you should have locked the revision during the previous check-out. Your last check-out should have been as follows:

co -l f.c

Of course, it is too late now to do the check-out with locking, because you probably modified *f.c* already, and a second check-out would overwrite your modifications. Instead, invoke the following:

rcs -l f.c

This command will lock the latest revision for you, unless somebody else got ahead of you already. In this case, you'll have to negotiate with that person.

Locking assures that you, and only you, can check in the next update, and avoids nasty problems if several people work on the same file. Even if a revision is locked, it can still be checked out for reading, compiling, etc. All that locking prevents is a check-in by anybody but the locker.

If your RCS file is private, i.e., if you are the only person who is going to deposit revisions into it, strict locking is not needed and you can turn it off. If strict locking is turned off, the owner of the RCS file need not have a lock for check-in; all others still do. Turning strict locking off and on is done with these commands:

rcs -U f.c and **rcs -L f.c**

If you don't want to clutter your working directory with RCS files, create a subdirectory called *RCS* in your working directory and move all your RCS files there. RCS commands will look first in that directory to find needed files. All the commands discussed above will still work, without any modification. (Actually, pairs of RCS and working files can be specified in three ways: (a) both are given, (b) only the working file is given, (c) only the RCS file is given.

Both RCS and working files may have arbitrary path prefixes; RCS commands pair them up intelligently).

To avoid the deletion of the working file during check-in (in case you want to continue editing), invoke the following command:

ci –l f.c or **ci –u f.c**

These commands check in *f.c* as usual, but perform an implicit check-out. The first form also locks the checked in revision, the second one doesn't. Thus, these options save you one check-out operation. The first form is useful if locking is strict, the second one if not strict. Both update the identification markers in your working file (see below).

You can give **ci** the number you want assigned to a checked in revision. Assume all your revisions were numbered 1.1, 1.2, 1.3, etc., and you would like to start release 2. The following command assigns the number 2.1 to the new revision.

ci –r2 f.c or **ci –r2.1 f.c**

From then on, **ci** will number the subsequent revisions with 2.2, 2.3, and so on. The corresponding **co** commands retrieve the latest revision numbered 2.x and the revision 2.1, respectively.

co –r2 f.c and **co –r2.1 f.c**

co without a revision number selects the latest revision on the "trunk" (i.e., the highest revision with a number consisting of 2 fields). Numbers with more than 2 fields are needed for branches. For example, to start a branch at revision 1.3, invoke this command:

ci –r1.3.1 f.c

This command starts a branch numbered 1 at revision 1.3 and assigns the number 1.3.1.1 to the new revision. For more information about branches, see **rcsfile**(4).

Automatic Identification

RCS can put special strings for identification into your source and object code. To obtain such identification, place the following marker into your text, for instance inside a comment.

$Header$

RCS will replace this marker with a string of this form:

$Header: filename revision_number date time author state $

With such a marker on the first page of each module, you can always see with which revision you are working. RCS keeps the markers up to date automatically. To propagate the markers into your object code, simply put them into literal character strings. In C, this is done as follows:

#ident "$Header$"

The command **ident** extracts such markers from any file, even object code and dumps. Thus, **ident** lets you find out which revisions of which modules were used in a given program.

You may also find it useful to put the marker Log into your text, inside a comment. This marker accumulates the log messages that are requested

during check-in. Thus, you can maintain the complete history of your file directly inside it. There are several additional identification markers; see **co**(1) for details.

SEE ALSO

ci(1), co(1), ident(1), merge(1), rcs(1), rcsdiff(1), rcsmerge(1), rlog(1)

NOTES

Copyright 1982 by Walter F. Tichy.

NAME

rcsmerge – merge RCS revisions

SYNOPSIS

rcsmerge **-r***rev1* [**-r***rev2*] [**-p**] **file**

DESCRIPTION

rcsmerge incorporates the changes between *rev1* and *rev2* of an RCS file into the corresponding working file. If **-p** is given, the result is printed on the standard output, otherwise the result overwrites the working file.

A filename ending in *,v* is an RCS filename, otherwise a working filename. **merge** derives the working filename from the RCS filename and vice versa, as explained in **co**(1). A pair consisting of both an RCS and a working filename may also be specified.

rev1 may not be omitted. If *rev2* is omitted, the latest revision on the default branch (normally the highest branch on the trunk) is assumed. Both *rev1* and *rev2* may be given numerically or symbolically.

rcsmerge prints a warning if there are overlaps, and delimits the overlapping regions as explained in **co -j**. The command is useful for incorporating changes into a checked-out revision.

EXAMPLES

Suppose you have released revision 2.8 of *f.c*. Assume furthermore that you just completed revision 3.4, when you receive updates to release 2.8 from someone else. To combine the updates to 2.8 and your changes between 2.8 and 3.4, put the updates to 2.8 into file *f.c* and execute the following command:

```
rcsmerge -p -r2.8 -r3.4 f.c >f.merged.c
```

Then examine **f.merged.c**. Alternatively, if you want to save the updates to 2.8 in the RCS file, check them in as revision 2.8.1.1 and execute **co -j**:

```
ci -r2.8.1.1 f.c
co -r3.4 -j2.8:2.8.1.1 f.c
```

As another example, the following command undos the changes between revision 2.4 and 2.8 in your currently checked out revision in *f.c*.

```
rcsmerge -r2.8 -r2.4 f.c
```

Note the order of the arguments, and that *f.c* will be overwritten.

SEE ALSO

ci(1), co(1), merge(1), ident(1), rcs(1), rcsdiff(1), rlog(1)

NOTES

Copyright 1982 by Walter F. Tichy.

BUGS

rcsmerge does not work on files containing lines with a single '.'.

NAME

readfile, longline – reads file and gets longest line

SYNOPSIS

readfile file
longline [file]

DESCRIPTION

The **readfile** command reads the file named in its argument. No translation of newlines is done. It keeps track of the longest line it reads and if there is a subsequent call to **longline**, the length of that line, including the newline character, is returned. The **longline** command can be given an argument instead, in which case it returns the longest line of the argument *file*.

EXAMPLES

Here is a typical use of **readfile** and **longline** in a text object:

```
.
.
.
text="`readfile myfile`"
columns=`longline`
.
.
.
```

DIAGNOSTICS

If the file does not exist, readfile will return FALSE (that is, the expression will have an error return).

SEE ALSO

cat(1)

NAME

regcmp – regular expression compile

SYNOPSIS

regcmp [–] files

DESCRIPTION

The **regcmp** command performs a function similar to **regcmp**(3X) and, in most cases, precludes the need for calling **regcmp**(3X) from C programs. This saves on both execution time and program size. The command **regcmp** compiles the regular expressions in *file* and places the output in *file.i*. If the – option is used, the output will be placed in *file.c*. The format of entries in *file* is a name (C variable) followed by one or more blanks followed by a regular expression enclosed in double quotes. The output of **regcmp** is C source code. Compiled regular expressions are represented as **extern char** vectors. *file.i* files may thus be included in C programs, or *file.c* files may be compiled and later loaded. In the C program which uses the **regcmp** output, **regex**(**abc**,**line**) will apply the regular expression named *abc* to *line*. Diagnostics are self-explanatory.

EXAMPLES

name "([A–Za–z][A–Za–z0–9_]*)$0"

telno "\({0,1}([2–9][01][1–9])$0\){0,1} *"
"([2–9][0–9]{2})$1[–]{0,1}"
"([0–9]{4})$2"

In the C program that uses the **regcmp** output, the following line will apply the regular expression named *telno* to *line*.

```
regex(telno, line, area, exch, rest)
```

SEE ALSO

regcmp(3X)

NAME

regex – match patterns against a string

SYNOPSIS

| **regex** [**–e**] [pattern template]...pattern [template]
regex [**–e**] **–v** "string" [pattern template]...pattern [template]

DESCRIPTION

The **regex** command takes a string (from *stdin*, or supplied with **–v**) and a list of *pattern/template* pairs, and runs **regex**(3X) on the string against each of the patterns until there is a match. When a match occurs, it writes the corresponding *template* to *stdout* and returns TRUE. The last (or only) *pattern* does not need a *template*. If the last or only *pattern* matches the string, the function returns TRUE. If no match is found, **regex** returns FALSE.

The **–e** option tells the function to evaluate the corresponding *template* and write the result to *stdout*.

The patterns are regular expressions of the form described in **regex**(3X). In most cases, the pattern should be enclosed in single quotes to turn off special meanings of characters.

The *template* may contain the strings $m0 through $m9, which are expanded to the part of the *pattern* enclosed in (...)$0 through (...)$9 constructs (see examples below). Note that if you use this feature, you must be sure to enclose the *template* in single quotes so that the Interpreter doesn't expand the $m0 through $m9 variables at parse time. This feature gives **regex** much of the power of **cut**(1), **paste**(1), and **grep**(1),and some of the capabilities of **sed**(1). If there is no *template*, the default is as follows:

"$m0$m1$m2$m3$m4$m5$m6$m7$m8$m9"

Note that only the final *pattern* may lack a *template*.

EXAMPLES

To cut the 4th through 8th letters out of a string:

```
'regex -v "my string is nice" '^.{3}(.{5})$0' '$m0'
```

In a form, for validating input as an integer:

```
valid='regex -v "$F5" '^[0-9]+$'
```

In a form, to translate an environment variable that contains one of the numbers 1, 2, 3, 4, 5 to the letters a, b, c, d, e:

```
value='regex -v "$VAR1" 1 a 2 b 3 c 4 d 5 e '.*'
'Error'
```

Note the use of the pattern '.*' to mean anything else.

In the example below, all three lines constitute a single backquoted expression. This expression, by itself, could be put in a menu descriptor file. Since backquoted expressions are expanded as they are parsed and output from a backquoted expression (the **cat** command, in this example) becomes part of the descriptor file being parsed, this expression would read */etc/passwd* and make a virtual menu of all the login ids on the system.

```
'cat /etc/passwd | regex '^([^:]*)$0.*$' ' name=$m0
action='message "$m0 is a user"'
```

SEE ALSO

cut(1), grep(1), paste(1), sed(1), regcmp(3)

DIAGNOSTICS

If none of the patterns match, **regex** returns FALSE, otherwise TRUE.

WARNINGS

Patterns and templates must usually be enclosed in single quotes to turn off the special meanings of characters, especially if you use the $m0 through $m9 variables in the template, since the Interpreter expands the variables (usually to "") before **regex** even sees them.

BUGS

The regular expressions accepted by *regcmp* differ slightly from other utilities such as **sed**, **grep**, **awk**, and **ed**.

Regex with the **–e** option forces subsequent commands to be ignored. In other words, if a backquoted statement appears as

```
'regex -e ...; command1; command2'
```

command1 and *command2* would never be executed. However, dividing the expression into two, as shown below, would yield the desired result.

```
'regex -e ...''command1; command2'
```

NAME

reinit – run an initialization file

SYNOPSIS

reinit init_file

DESCRIPTION

The **reinit** command takes an initialization file name as its only argument. The Interpreter will parse and execute this file and then continue running the current application. **reinit** is typically used to change the defaults set by the initialization file that was named when **fmli** was invoked.

The **reinit** command does not re-display the introductory object or change the Screen-labeled Keys layout.

NAME

repquota – summarize quotas for a filesystem

SYNOPSIS

/usr/etc/repquota [**–v**] *filesystem...*

/usr/etc/repquota [**–v**] **–a**

DESCRIPTION

repquota displays a summary of the disk usage and quotas for the specified filesystems. It shows the current number of files and the amount of space (in kilobytes) for each user, along with any quotas created with **edquota**(1M).

The ability of users to view other user's quotas, is determined by the protection of the *quotas* file, at the root of each filesystem. Normally, the protection of this file is set so that only the superuser can view another user's quotas.

The following options are available:

–a Report on all filesystems whose entries in */etc/fstab* indicate that they allow read-write access and have quotas.

–v Include all users having quotas, even if they are not using space.

FILES

quotas	At the root of each filesystem with quotas
/etc/fstab	Has filesystem names and locations

SEE ALSO

edquota(1M), quota(1), quotacheck(1M), quotaon(1M), quotactl(2seq)

DIAGNOSTICS

Various self-explanatory messages about inaccessible files

NAME

reset – reset the current form field to its default values

SYNOPSIS

reset

DESCRIPTION

The **reset** command resets a field in a form to its default value — the value displayed when the form was first opened.

NAME

restore – incremental filesystem restore

SYNOPSIS

/etc/restore key [name ...]

DESCRIPTION

restore reads tapes dumped with the **dump**(1M) command. Its actions are controlled by the *key* argument. The *key* is a string of characters containing at most one function letter and possibly one or more function modifiers. Other arguments to the command are file or directory names specifying the files that are to be restored. Unless the **h** key is specified (see below), the appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

The function portion of the key is specified by one of the following letters:

r The tape is read and loaded into the current directory. This should not be done lightly; the **r** key should only be used to restore a complete dump tape onto a clear filesystem or to restore an incremental dump tape after a full level zero restore. Thus, the following is a typical sequence to restore a complete dump.

```
/etc/newfs  /dev/rdsk/zd0s3
/etc/mount  /dev/rdsk/zd0s3  /mnt
cd  /mnt
restore  r
```

Another **restore** can be done to get an incremental dump in on top of this. Note that **restore** leaves a file *restoresymtab* in the root directory to pass information between incremental restore passes. This file should be removed when the last incremental tape has been restored.

A **dump**(1M) followed by a **mkvtoc**(1M), a **newfs**(1M), and a **restore** is used to change the size of a filesystem.

R **restore** requests a particular tape of a multi-volume set on which to restart a full restore (see the **r** key above). This allows **restore** to be interrupted and then restarted.

x The named files are extracted from the tape. If the named file matches a directory whose contents were written onto the tape and the **h** key is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). If no file argument is given, the root directory is extracted, which results in the entire contents of the tape being extracted (unless the **h** key was specified).

t The names of the specified files are listed if they occur on the tape. If no file argument is given, the root directory is listed, which results in the entire contents of the tape being listed (unless the **h** key was specified).

i This mode allows interactive restoration of files from a dump tape. After reading in the directory information from the tape, **restore** provides a shell-like interface that allows the user to move around the directory tree selecting files to be extracted. The available commands are given below; for those commands that require an argument, the default is the current directory.

ls [arg] – List the current or specified directory. Entries that are directories are appended with a slash (/). Entries that have been marked for extraction are prepended with a asterisk (*). If the verbose key is set, the inode number of each entry is also listed.

cd arg – Change the current working directory to the specified argument.

pwd – Print the full pathname of the current working directory.

add [arg] – The current directory or specified argument is added to the list of files to be extracted. If a directory is specified, then it and all its descendents are added to the extraction list (unless the **h** key is specified on the command line). Files that are on the extraction list are prepended with an asterisk (*) when they are listed by ls.

delete [arg] – The current directory or specified argument is deleted from the list of files to be extracted. If a directory is specified, then it and all its descendents are deleted from the extraction list (unless the **h** key is specified on the command line). The most expedient way to extract most of the files from a directory is to add the directory to the extraction list and then delete those files that are not needed.

extract – All the files that are on the extraction list are extracted from the dump tape. **restore** will ask which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume and work towards the first volume.

verbose – The sense of the **v** key is toggled. When set, the verbose key causes the **ls** command to list the inode numbers of all entries. It also causes **restore** to print out information about each file as it is extracted.

help – List a summary of the available commands.

quit – **restore** immediately exits, even if the extraction list is not empty.

The following characters may be used in addition to the letter that selects the function desired.

v Normally **restore** does its work silently. The **v** (verbose) key causes it to print the name of each file it treats preceded by its file type.

f The next argument to **restore** is used as the name of the archive instead of */dev/rmt/xt0h*. If the name of the file is "–", **restore** reads from standard input. Thus, **dump**(1M) and **restore** can be used in a pipeline to dump and restore a filesystem with this command:

dump 0f – /usr | (cd /mnt; restore xf –)

y **restore** will not ask whether it should abort the restore if gets a tape error. It will always try to skip over the bad tape block(s) and continue as best it can.

m **restore** will extract by inode numbers rather than by filename. This is useful if only a few files are being extracted and you want to avoid regenerating the complete pathname to the file.

h **restore** extracts the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the tape.

s The next argument to **restore** designates the number of files to skip from the beginning of the tape before using that tape for processing. This allows multiple (short) dumps to reside on a single tape, separated by logical EOF marks. If not given, **restore** begins processing with the first information right after the physical BOT label.

b The next argument to **restore** designates the blocksize (in 1024 byte units) to read the tape. This option is useful when performing restores where tape is an actual file in the filesystem or a streamer tape where the interface acts like a block interface. **restore** attempts to intuit the blocksize by looking at the return from the first read of the tape. **dump**(1M) writes **NTREC** block records (currently 10) to normal density tapes unless given a different blocksize with the **b** option. Higher density tapes (such as 6250 bpi) are written with **HIGHDENSITYTREC** block records (currently 32).

Note: DYNIX (version 3.0 or earlier) conditional symbolic links on the dump tape are restored, but only the "ucb" part of the link is restored.

FILES

/dev/rmt/xt0h	The default tape drive
/tmp/rstdir*	File containing directories on the tape
/tmp/rstmode*	Owner, mode, and time stamps for directories
./restoresymtab	Information passed between incremental restores

SEE ALSO

cpio(1), tar(1), dump (1M), newfs(1M), mkfs(1M), mount(1M), volcopy(1M)

DIAGNOSTICS

Complains about bad key characters.

Complains if it gets a read error. If **y** has been specified or the user responds "y", **restore** will attempt to continue the restore.

If the dump extends over more than one tape, **restore** will ask the user to change tapes. If the **x** or **i** key has been specified, **restore** will also ask which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume and work towards the first volume.

There are numerous consistency checks that can be listed by **restore**. Most checks are self-explanatory or can "never happen." Common errors are given below.

Converting to new file system format

A dump tape created from the old filesystem has been loaded. It is

automatically converted to the new filesystem format.

<filename>: not found on tape
The specified filename was listed in the tape directory, but was not found on the tape. This is caused by tape read errors while looking for the file, and from using a dump tape created on an active filesystem.

expected next file <inumber>, got <inumber>
A file that was not listed in the directory showed up. This can occur when using a dump tape created on an active filesystem.

Incremental tape too low
When doing incremental restore, a tape that was written before the previous incremental tape or that has too low an incremental level has been loaded.

Incremental tape too high
When doing incremental restore, a tape that does not begin its coverage where the previous incremental tape left off or that has too high an incremental level has been loaded.

Tape read error while restoring <filename>
Tape read error while skipping over inode <inumber>
Tape read error while trying to resynchronize
A tape read error has occurred. If a filename is specified, then its contents are probably partially wrong. If an inode is being skipped or the tape is trying to resynchronize, then no extracted files have been corrupted, though files may not be found on the tape.

resync restore, skipped <num> blocks
After a tape read error, **restore** may have to resynchronize itself. This message lists the number of blocks that were skipped over.

This error can also occur if the blocksize used to read the tape does not agree with the blocksize used to write it. This commonly occurs when tape is actually a file in the filesystem or a streamer tape where the write blocksize is not preserved. File damage has occured if you see this message. Try using the **b** key with the correct blocksize to overcome this error.

BUGS

restore can get confused when doing incremental restores from dump tapes that were made on active filesystems.

When reading from tape, **restore** reads until it needs no more data, not until tape EOF. For multiple dumps saved on the same tape, you will need to advance the tape manually past the EOF marker after each restore is complete to get to the next dump (with "mt fsf 1").

A level zero dump must be done after a full restore. Because restore runs in user code, it has no control over inode allocation; thus a full dump must be done to get a new set of directories reflecting the new inode numbering, even though the contents of the files is unchanged.

NAME

rlog – print log messages and other information about RCS files

SYNOPSIS

rlog [options] file ...

DESCRIPTION

rlog prints information about RCS files. Files ending in *,v* are RCS files, all others are working files. If a working file is given, **rlog** tries to find the corresponding RCS file first in directory *./RCS* and then in the current directory, as explained in **co**(1).

rlog prints the following information for each RCS file: RCS filename, working filename, head (i.e., the number of the latest revision on the trunk), default branch, access list, locks, symbolic names, suffix, total number of revisions, number of revisions selected for printing, and descriptive text. This is followed by entries for the selected revisions in reverse chronological order for each branch. For each revision, **rlog** prints revision number, author, date/time, state, number of lines added/deleted (with respect to the previous revision), locker of the revision (if any), and log message. Without options, **rlog** prints complete information. The options below restrict this output.

–L Ignores RCS files that have no locks set; convenient in combination with **–R**, **–h**, or **–l**.

–R Prints only the name of the RCS file; convenient for translating a working filename into an RCS filename.

–h Prints only the RCS filename, working filename, head, default branch, access list, locks, symbolic names, and suffix.

–t Prints the same as **–h**, plus the descriptive text.

–b Prints information about the revisions on the default branch (normally the highest branch on the trunk).

–d*dates* Prints information about revisions with a check-in date/time in the ranges given by the semicolon-separated list of *dates*. A range of the form *d1<d2* or *d2>d1* selects the revisions that were deposited between *d1* and *d2*, (inclusive). A range of the form *<d* or *d>* selects all revisions dated *d* or earlier. A range of the form *d<* or *>d* selects all revisions dated *d* or later. A range of the form *d* selects the single, latest revision dated *d* or earlier. The date/time strings *d, d1,* and *d2* are in the free format explained in **co**(1). Quoting is normally necessary, especially for < and >. Note that the separator is a semicolon.

–l*[lockers]* Prints information about locked revisions. If the comma-separated list *lockers* of login names is given, only the revisions locked by the given login names are printed. If the list is omitted, all locked revisions are printed.

–r*revisions* Prints information about revisions given in the comma-separated list *revisions* of revisions and ranges. A range *rev1-rev2* means revisions *rev1* to *rev2* on the same branch, *–rev* means revisions from the beginning of the branch up to and including *rev*, and *rev–* means revisions starting with *rev* to the end of the branch

containing *rev*. An argument that is a branch means all revisions on that branch. A range of branches means all revisions on the branches in that range.

-s*states* Prints information about revisions whose state attributes match one of the states given in the comma-separated list *states*.

-w*[logins]* Prints information about revisions checked in by users with login names appearing in the comma-separated list *logins*. If *logins* is omitted, the user's login is assumed.

rlog prints the intersection of the revisions selected with the options **-d**, **-l**, **-s**, **-w**, intersected with the union of the revisions selected by **-b** and **-r**.

EXAMPLES

```
rlog -L -R RCS/*,v
rlog -L -h RCS/*,v
rlog -L -l RCS/*,v
rlog RCS/*,v
```

The first command prints the names of all RCS files in the subdirectory *RCS* that have locks. The second command prints the headers of those files, and the third prints the headers plus the log messages of the locked revisions. The last command prints complete information.

SEE ALSO

ci(1), co(1), ident(1), rcs(1), rcsdiff(1), rcsintro(1), rcsmerge(1), rcsfile(4)

NOTES

Copyright 1982 by Walter F. Tichy.

DIAGNOSTICS

The exit status always refers to the last RCS file operated on and is 0 if the operation was successful, 1 otherwise.

NAME

rm, rmdir – remove files or directories

SYNOPSIS

rm [**–f**] [**–i**] file ...

rm –r [**–f**] [**–i**] dirname ... [file ...]

rmdir [**–p**] [**–s**] dirname ...

DESCRIPTION

rm removes the entries for one or more files from a directory. If an entry was the last link to the file, the file is destroyed. If the tacky bit is not set on a directory, removal of a file in that directory requires write permission on the directory, but neither read nor write permission on the file itself. If the tacky bit is set, you must have write permission for both the directory and the file you want to remove.

If a file has no write permission and the standard input is a terminal, the full set of permissions (in octal) for the file are printed followed by a question mark. This is a prompt for confirmation. If the answer begins with **y** (for yes), the file is deleted, otherwise the file remains.

Note that if the standard input is not a terminal, the command will operate as if the **–f** option is in effect.

rmdir removes the named directories, which must be empty.

Three options apply to **rm**:

–f This option causes the removal of all files (whether write-protected or not) in a directory without prompting the user. In a write-protected directory, however, files are never removed (whatever their permissions are), but no messages are displayed. If the removal of a write-protected directory was attempted, this option cannot suppress an error message.

–r This option causes the recursive removal of any directories and subdirectories in the argument list. The directory will be emptied of files and removed. Note that the user is normally prompted for removal of any write-protected files which the directory contains. The write-protected files are removed without prompting, however, if the **–f** option is used, or if the standard input is not a terminal and the **–i** option is not used.

If the removal of a non-empty, write-protected directory was attempted, the command will always fail (even if the **–f** option is used), resulting in an error message.

–i With this option, confirmation of removal of any write-protected file occurs interactively. It overrides the **–f** option and remains in effect even if the standard input is not a terminal.

Two options apply to **rmdir**:

–p This option allows users to remove the directory *dirname* and its parent directories which become empty. A message is printed on standard output as to whether the whole path is removed or part of the path remains for some reason.

–s This option is used to suppress the message printed on standard error when **–p** is in effect.

SEE ALSO

unlink(2), rmdir(2)

DIAGNOSTICS

All messages are generally self-explanatory.

It is forbidden to remove the files "." and ".." in order to avoid the consequences of inadvertently doing something like the following:

```
rm -r .*
```

Both **rm** and **rmdir** return exit codes of 0 if all the specified directories are removed successfully. Otherwise, they return a non-zero exit code.

NAME

rmdel – remove a delta from an SCCS file

SYNOPSIS

rmdel –rSID files

DESCRIPTION

rmdel removes the delta specified by the *SID* from each named SCCS file. The delta to be removed must be the newest (most recent) delta in its branch in the delta chain of each named SCCS file. In addition, the release specified must not be that of a version being edited for the purpose of making a delta (that is, if a *p-file* [see **get**(1)] exists for the named SCCS file, the release specified must not appear in any entry of the *p-file*).

The **–r** option is used for specifying the *SID* (SCCS Identification) level of the delta to be removed.

If a directory is named, **rmdel** behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the pathname does not begin with *s.*) and unreadable files are silently ignored. If a name of – is given, the standard input is read. Each line of the standard input is taken to be the name of an SCCS file to be processed; non-SCCS files and unreadable files are silently ignored.

Simply stated, (1) if you make a delta you can remove it; or (2) if you own the file and directory you can remove a delta.

FILES

x.file	[See **delta**(1)]
z.file	[See **delta**(1)]

SEE ALSO

delta(1), get(1), help(1), prs(1), sccsfile(4)

DIAGNOSTICS

Use *help*(1) for explanations.

NAME

rmvopt – remove a menu option

SYNOPSIS

rmvopt –M *menu-name* **–t** *task-name* [**–F**]

DESCRIPTION

This command removes an option from a menu on the ptx/ADMIN menu system.

M *parentmenu* The name of the menu that displays the option. For example, **c1print**.

d *taskname* The name of task that the option calls. For example, the name of the task for the Display Outstanding Calls task is **dspcalls**.

–F Use the **–F** option to delete all of the files that support the task. If you do not select this option, the system removes the option from its *parentmenu*, but related files will remain on the system.

Environment Variables

TFADMFILE Identifies the directory where the ptx/ADMIN administrative data files are located.

FILES

${TFADMFILE}/etc/task
Task table

${TFADMFILE}/menus/chap1/*parentmenu*
Menu displaying the option

SEE ALSO

addcmdopt(1M), addformopt(1M), addmenuopt(1M)

NAME

run – run an executable

SYNOPSIS

run [**–s**] [**–e**] [**–n**] [**–t** title] program

DESCRIPTION

The **run** command runs a program, using the**$PATH** environment variable to find it. The **–s** option means silent, meaning that the screen is not redrawn when this is done. The **–e** option prompts the user before returning to the Interpreter only if there is an error condition (by default the user is always prompted). The **–n** option never prompts the user (useful for programs like **vi** which the user must do some specific action to exit anyway). The **–t** option gives the name this process will have in the pop-up menu generated by the **frm-list** command. This option implies the ability to suspend the operating system process and return to the FMLI application.

EXAMPLES

Here is a menu that uses **run:**

```
menu="Edit special System files"

name="Password file"
action='run -e vi /etc/passwd'

name="Group file"
action='run -e vi /etc/group'

name="My .profile"
action='run -n vi $HOME/.profile'
```

NAME

runacct – run daily accounting

SYNOPSIS

/usr/lib/acct/runacct [mmdd [state]]

DESCRIPTION

runacct is the main daily accounting shell procedure. It is normally initiated via **cron**(1M). **runacct** processes connect, fee, disk, and process accounting files. It also prepares summary files for **prdaily** or billing purposes.

runacct takes care not to damage active accounting files or summary files in the event of errors. It records its progress by writing descriptive diagnostic messages into *active*. When an error is detected, a message is written to */dev/console*, mail (see **mail**(1)) is sent to **root** and **adm**, and **runacct** terminates. **runacct** uses a series of lock files to protect against re-invocation. The files *lock* and *lock1* are used to prevent simultaneous invocation, and *lastdate* is used to prevent more than one invocation per day.

runacct breaks its processing into separate, restartable *states* using *statefile* to remember the last *state* completed. It accomplishes this by writing the *state* name into *statefile*. **runacct** then looks in *statefile* to see what it has done and to determine what to process next. *states* are executed in the following order:

SETUP	Move active accounting files into working files.
WTMPFIX	Verify integrity of *wtmp* file, correcting date changes if necessary.
CONNECT1	Produce connect session records in *ctmp.h* format.
CONNECT2	Convert *ctmp.h* records into *tacct.h* format.
PROCESS	Convert process accounting records into *tacct.h* format.
MERGE	Merge the connect and process accounting records.
FEES	Convert output of *chargefee* into *tacct.h* format and merge with connect and process accounting records.
DISK	Merge disk accounting records with connect, process, and fee accounting records.
MERGETACCT	Merge the daily total accounting records in *daytacct* with the summary total accounting records in */usr/adm/acct/sum/tacct*.
CMS	Produce command summaries.
USEREXIT	Any installation-dependent accounting programs can be included here.
CLEANUP	Cleanup temporary files and exit.

To restart **runacct** after a failure, first check the *active* file for diagnostics, then fix up any corrupted data files such as *pacct* or *wtmp*. The *lock* files and *lastdate* file must be removed before **runacct** can be restarted. The argument *mmdd* is necessary if **runacct** is being restarted. It specifies the month and day for which **runacct** will rerun the accounting. Entry point for processing is based on the contents of *statefile*. To override this, include the desired *state*

on the command line to designate where processing should begin.

EXAMPLES

To start **runacct**:

nohup runacct 2> /usr/adm/acct/nite/fd2log &

To restart **runacct**:

nohup runacct 0601 2>> /usr/adm/acct/nite/fd2log &

To restart **runacct** at a specific *state*:

nohup runacct 0601 MERGE 2>> /usr/adm/acct/nite/fd2log &

FILES

/etc/wtmp
/usr/adm/pacct*
/usr/src/cmd/acct/tacct.h
/usr/src/cmd/acct/ctmp.h
/usr/adm/acct/nite/active
/usr/adm/acct/nite/daytacct
/usr/adm/acct/nite/lock
/usr/adm/acct/nite/lock1
/usr/adm/acct/nite/lastdate
/usr/adm/acct/nite/statefile
/usr/adm/acct/nite/ptacct*.*mmdd*

SEE ALSO

acctcom(1), mail(1), acct(1M), acctcms(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), cron(1M), fwtmp(1M), acct(2), acct(4), utmp(4)

BUGS

Normally it is not a good idea to restart **runacct** in the **SETUP** *state*. Run **SETUP** manually and restart via:

runacct *mmdd* **WTMPFIX**

If **runacct** failed in the **PROCESS** *state,* remove the last *ptacct* file.

NAME

sact – print current SCCS file editing activity

SYNOPSIS

sact files

DESCRIPTION

sact informs the user of any impending deltas to a named SCCS file. This situation occurs when **get**(1) with the **–e** option has been previously executed without a subsequent execution of **delta**(1). If a directory is named on the command line, **sact** behaves as though each file in the directory were specified as a named file, except that non-SCCS files and unreadable files are silently ignored. If a name of – is given, the standard input is read with each line being taken as the name of an SCCS file to be processed.

The output for each named file consists of five fields separated by spaces.

Field 1	Specifies the SID of a delta that currently exists in the SCCS file to which changes will be made to make the new delta.
Field 2	Specifies the SID for the new delta to be created.
Field 3	Contains the logname of the user who will make the delta (that is, who executed a **get** for editing).
Field 4	Contains the date that **get –e** was executed.
Field 5	Contains the time that **get –e** was executed.

SEE ALSO

delta(1), get(1), unget(1)

DIAGNOSTICS

Use **help**(1) for explanations.

NAME

sag – system activity graph

SYNOPSIS

sag [options]

DESCRIPTION

sag graphically displays the system activity data stored in a binary data file by a previous **sar**(1) run. Any of the **sar** data items can be plotted singly or in combination; as cross plots, or versus time. Simple arithmetic combinations of data may be specified. **sag** invokes **sar** and finds the desired data by string-matching the data column header (run **sar** to see what is available). These *options* are passed through to **sar**:

–s *time* Select data later than *time* in the form hh[:mm]. Default is 08:00.

–e *time* Select data up to *time*. Default is 18:00.

–i *sec* Select data at intervals as close as possible to *sec* seconds.

–f *file* Use *file* as the data source for **sar**. Default is the current daily data file */usr/adm/sa/sadd*.

Other *options*:

–T *term* Produce output suitable for terminal *term*. See **tplot**(1G) for known terminals. Default for *term* is **$TERM**.

–x *spec* x axis specification with *spec* in the following form:

"name[op name]...[lo hi]"

–y *spec* y axis specification with *spec* in the same form as above.

name is either a string that will match a column header in the **sar** report with an optional device name in square brackets (such as **r+w/s[dsk–1]**) or an integer value. *Op* is + – * or / surrounded by blanks. Up to five names may be specified. Parentheses are not recognized. Contrary to custom, + and – have precedence over * and /. Evaluation is left to right. Thus A / A + B * 100 is evaluated (A/(A+B))*100, and A + B / C + D is (A+B)/(C+D). *lo* and *hi* are optional numeric scale limits. If unspecified, they are deduced from the data.

A single *spec* is permitted for the x axis. If unspecified, *time* is used. Up to 5 *spec*'s separated by semicolons (;) may be given for **–y**. Enclose the **–x** and **–y** arguments in quotes (" ") if they include whitespace. The **–y** default is:

–y "%usr 0 100; %usr + %sys 0 100; %usr + %sys + %wio 0 100"

EXAMPLES

To see today's CPU utilization, type this command:

```
sag
```

To see activity over 15 minutes of all disk drives:

```
TS=`date +%H:%M`
sar –o tempfile 60 15
TE=`date +%H:%M`
sag –f tempfile –s $TS –e $TE –y "r+w/s[dsk]"
```

FILES

/usr/adm/sa/sa*dd* Daily data file for day *dd*

SEE ALSO

sar(1), tplot(1G)

NAME

sar – system activity reporter

SYNOPSIS

sar [**–ubdycwaqvmprA**] [**–o** file] t [n]

sar [**–ubdycwaqvmprA**] [**–s** time] [**–e** time] [**–i** sec] [**–f** file]

DESCRIPTION

sar, in the first instance, samples cumulative activity counters in the operating system at *n* intervals of *t* seconds, where *t* should be 5 or greater. If the **–o** option is specified, it saves the samples in *file* in binary format. The default value of *n* is 1. In the second instance, with no sampling interval specified, **sar** extracts data from a previously recorded *file*, either the one specified by the **–f** option or, by default, the standard system activity daily data file */usr/adm/sa/sa dd* for the current day *dd*. The starting and ending times of the report can be bounded via the **–s** and **–e** *time* arguments of the form *hh*[*:mm*[*:ss*]]. The **–i** option selects records at *sec* second intervals. Otherwise, all intervals found in the data file are reported.

In either case, subsets of data to be printed are specified by option:

–u Report CPU utilization (the default):

%usr, %sys, %wio, %idle – portion of time running in user mode, running in system mode, idle with some process waiting for block I/O, and otherwise idle.

–b Report buffer activity:

bread/s, bwrit/s – transfers per second of data between system buffers and disk or other block devices;

lread/s, lwrit/s – accesses of system buffers;

%rcache, %wcache – cache hit ratios, i. e., (1–bread/lread) as a percentage;

pread/s, pwrit/s – transfers via raw (physical) device mechanism.

–d Report activity for each block device such as a disk or tape drive. When data is displayed, the device specification *dsk–* is generally used to represent a disk drive. The device specification used to represent a tape drive is machine dependent. The following activity data is reported:

%busy, avque – portion of time device was busy servicing a transfer request, average number of requests outstanding during that time;

r+w/s, blks/s – number of data transfers from or to device, number of bytes transferred in 512-byte units;

avwait, avserv – average time in ms. that transfer requests wait idly on queue, and average time to be serviced (which for disks includes seek, rotational latency and data transfer times).

–y Report TTY device activity:

rawch/s, canch/s, outch/s – input character rate, input character rate processed by canon, output character rate;

rcvin/s, xmtin/s, mdmin/s – receive, transmit and modem interrupt rates.

-c Report system calls:

scall/s – system calls of all types;

sread/s, swrit/s, fork/s, exec/s – specific system calls;

rchar/s, wchar/s – characters transferred by read and write system calls.

-w Report system swapping and switching activity:

swpin/s, swpot/s, bswin/s, bswot/s – number of transfers and number of 512-byte units transferred for swapins and swapouts (including initial loading of some programs);

pswch/s – process switches.

-a Report use of file access system routines:

iget/s, namei/s, dirblk/s.

-q Report average queue length while occupied, and % of time occupied:

runq-sz, %runocc – run queue of processes in memory and runnable;

swpq-sz, %swpocc – swap queue of processes swapped out but ready to run.

-v Report status of process, inode, file tables:

text-sz, proc-sz, inod-sz, file-sz, lock-sz – entries/size for each table, evaluated once at sampling point;

ov – overflows that occur between sampling points for each table.

-m Report message and semaphore activities:

msg/s, sema/s – primitives per second.

-p Report paging activities:

vflt/s – address translation page faults (valid page not in memory);

pflt/s – page faults from protection errors (illegal access to page) or "copy-on-writes" (always 0 under DYNIX);

pgfil/s – vflt/s satisfied by page-in from filesystem;

rclm/s – valid pages reclaimed for free list (always 0 under DYNIX).

-r Report unused memory pages and disk blocks:

freemem – average pages available to user processes;

freeswap – disk blocks available for process swapping.

-A Report all data. Equivalent to **-udqbwcayvmpr**.

EXAMPLES

To see today's CPU activity so far:

```
sar
```

To watch CPU activity evolve for 10 minutes and save data:

```
sar -o temp 60 10
```

To later review disk and tape activity from that period:

sar –d –f temp

FILES

/usr/adm/sa/sa*dd* Daily data file, where *dd* are digits representing the day of the month.

SEE ALSO

sag(1G), sar(1M)

NAME

sar: sa1, sa2, sadc – system activity report package

SYNOPSIS

/usr/lib/sa/sadc [t n] [ofile]

/usr/lib/sa/sa1 [t n]

/usr/lib/sa/sa2 [**–ubdycwaqvmprA**] [**–s** time] [**–e** time] [**–i** sec]

DESCRIPTION

System activity data can be accessed at the special request of a user (see **sar**(1)) and automatically on a routine basis as described here. The operating system contains a number of counters that are incremented as various system actions occur. These include counters for CPU utilization, buffer usage, disk and tape I/O activity, TTY device activity, switching and system-call activity, file-access, queue activity, inter-process communications, and paging.

sadc and shell procedures, **sa1** and **sa2**, are used to sample, save, and process this data.

sadc, the data collector, samples system data *n* times every *t* seconds and writes in binary format to *ofile* or to standard output. The sampling interval *t* should be greater than 5 seconds; otherwise, the activity of **sadc** itself may affect the sample. If *t* and *n* are omitted, a special record is written. This facility is used at system boot time, when booting to a multiuser state, to mark the time at which the counters restart from zero. For example, the */etc/init.d/perf* file writes the restart mark to the daily data by the command entry:

```
su sys -c "/usr/lib/sa/sadc /usr/adm/sa/sa`date +%d`"
```

The shell script **sa1**, a variant of **sadc**, is used to collect and store data in binary file */usr/adm/sa/sadd* where *dd* is the current day. The arguments **t** and **n** cause records to be written *n* times at an interval of *t* seconds, or once if omitted. The following entries in */usr/spool/cron/crontabs/sys* (see **cron**(1M)) will produce records every 20 minutes during working hours and hourly otherwise.

```
0 * * * 0-6 /usr/lib/sa/sa1
20,40 8-17 * * 1-5 /usr/lib/sa/sa1
```

The shell script **sa2**, a variant of **sar**(1), writes a daily report in file */usr/adm/sa/sardd*. The options are explained in **sar**(1). The following */usr/spool/cron/crontabs/sys* entry will report important activities hourly during the working day.

```
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 1200 -A
```

The structure of the binary daily data file consists of one word (int) which denotes the number of elements in the *devio* array actually written followed by zero or more copies of the *sa* structure defined as follows:

```
struct sa {
     struct sysinfo si;
     struct minfo mi;
     int  szinode; /* current size of inode table  */
     int  szfile;  /* current size of file table  */
     int  szproc;  /* current size of proc table  */
     int   szlckf; /* current size of file record header table */
     int   szlckr; /* current size of file record lock table */
     int  mszinode;/* size of inode table  */
     int  mszfile; /* size of file table  */
     int  mszproc; /* size of proc table  */
     int   mszlckf;/* maximum size of file record header table */
     int   mszlckr;/* maximum size of file record lock table */
     long  inodeovf;      /* cumulative overflows of inode table  */
     long  fileovf;/* cumulative overflows of file table  */
     long  procovf;/* cumulative overflows of proc table  */
     time_t  ts;   /* time stamp, seconds  */
     struct dev_stats devio[NDEVS];/* device unit information  */
};
```

where:

```
struct sysinfo {
     time_t cpu[5];
#define CPU_IDLE    0
#define CPU_USER    1
#define CPU_KERNEL  2
#define CPU_WAIT    3
#define CPU_SXBRK   4
     time_t wait[3];
#define W_IO        0
#define W_SWAP      1
#define W_PIO       2
     long   bread;
     long   bwrite;
     long   lread;
     long   lwrite;
     long   phread;
     long   phwrite;
     long   swapin;
     long   swapout;
     long   bswapin;
     long   bswapout;
     long   pswitch;
     long   syscall;
     long   remcall;
     long   fileop;
     long   serve;
     long   sysread;
```

```
        long    syswrite;
        long    sysfork;
        long    sysexec;
        long    runque;
        long    runocc;
        long    swpque;
        long    swpocc;
        long    iget;
        long    namei;
        long    dirblk;
        long    readch;
        long    writech;
        long    rcvint;
        long    xmtint;
        long    mdmint;
        long    rawch;
        long    canch;
        long    outch;
        long    msg;
        long    sema;
        long    pnpfault;
        long    wrtfault;
};

struct minfo {
        unsigned long freemem[2]; /* freemem in pages */
                                  /* "double" long format*/
                                  /* freemem[0] least significant */
        long    freeswap;         /* free swap space */
        long    vfault;           /* translation fault */
        long    demand;           /*  demand zero and demand fill pages */
        long    swap;             /*  pages on swap */
        long    cache;            /*  pages in cache */
        long    file;             /*  pages on file */
        long    pfault;
        long    cw;
        long    steal;
        long    freedpgs;
        long    unmodsw;
        long    unmodfl;
};

struct dev_stats {
        char    d_name[12];     /* device name */
        ulong   d_ops;          /* number of I/o requests since boot */
        ulong   d_blks;         /* number of DEV_BSIZE blocks xferred */
        ulong   d_active;       /* cumulative time device was active */
                                /* in milliseconds */
        ulong   d_resp;         /* cumulative response time in milliseconds */
};
```

The following fields in **struct sysinfo** will always be 0: **remcall, fileop, serve, pnpfault,** and **wrtfault.** The following fields in **struct minfo** structure will always be 0: **pfault, cw, steal, freedpgs, unmodsw,** and **unmodfl.**

FILES

/usr/adm/sa/sa*dd*	Daily data file
/usr/adm/sa/sar*dd*	Daily report file
/tmp/sa_adrfl	Address file

SEE ALSO

cron(1M), sag(1G), sar(1), timex(1)

NAME

savecore – save a core dump of the operating system

SYNOPSIS

/etc/savecore *dirname* [*system*]

/etc/savecore –c

/etc/savecore –q

DESCRIPTION

savecore is meant to be called in */etc/init.d/savecore*. This file is hard-linked to */etc/rc2.d/<sequence_number>savecore*. Its function is to save the core dump of the system (assuming one was made).

savecore saves the core image in the file *dirname/vmcore.n*. (The default file is */usr/crash*.) If the core dump corresponds with the current running */unix* or the kernel specified in *system*, it is saved in *dirname/unix.n*. The trailing *.n* in the pathnames is replaced by a number that grows every time **savecore** is run in that directory. **savecore** also saves the output from **/etc/dmesg**(1M) into *dirname/dmesg.n*.

Before **savecore** writes out a core image, it reads a number from the file *dirname/minfree*. If there are fewer free blocks on the filesystem which contains *dirname* than the number obtained from the *minfree* file, the core dump is not done. If the *minfree* file does not exist, **savecore** always writes out the core file (assuming that a core dump was taken).

If the core dump was from a system other than */unix*, the name of that system must be supplied as *sysname*.

If **savecore** is invoked with **–c**, the dump, if one exists, is cleared (that is, discarded).

If **savecore** is invoked with **–q**, **savecore** checks to see if a dump exists. If a dump exists, `Dump exists` is printed. Otherwise, `No dump` is printed.

Warning: **Savecore** uses information in the disk's VTOC to determine how big the dump partition is. This implies that disks which are used for dump areas must have a VTOC present on them. See **mkvtoc**(1M) for information on adding a VTOC to the disk.

FILES

/unix Current operating system

SEE ALSO

mkvtoc(1M), **prtvtoc**(1M), **showcfg**(1M), **dump**(8)

NAME

sccsdiff – compare two versions of an SCCS file

SYNOPSIS

sccsdiff –rSID1 –rSID2 [–**p**] [–*sn*] files

DESCRIPTION

sccsdiff compares two versions of an SCCS file and generates the differences between the two versions. Any number of SCCS files may be specified, but arguments apply to all files.

–r*SID?* *SID1* and *SID2* specify the deltas of an SCCS file that are to be compared. Versions are passed to **bdiff**(1) in the order given.

–**p** Pipe output for each file through **pr**(1).

–*sn* *n* is the file segment size that **bdiff** will pass to **diff**(1). This is useful when **diff** fails due to a high system load.

FILES

/tmp/get????? Temporary files

SEE ALSO

bdiff(1), get(1), help(1), pr(1)

DIAGNOSTICS

"file : No differences" The two versions are the same.

Use **help**(1) for explanations.

NAME

sccstorcs – build RCS file from SCCS file

SYNOPSIS

sccstorcs [**–t**] [**–v**] *s.file* ...

DESCRIPTION

sccstorcs builds an RCS file from each SCCS file argument. The deltas and comments for each delta are preserved and installed into the new RCS file in order. Also preserved are the user access list and descriptive text, if any, from the SCCS file.

The following flags are meaningful:

–t Trace only. Prints detailed information about the SCCS file and lists the commands that would be executed to produce the RCS file. No commands are actually executed and no RCS file is made.

–v Verbose. Prints each command that is run while it is building the RCS file.

FILES

For each s.*somefile*, **sccstorcs** writes the files *somefile* and *somefile,v* which should not already exist. **sccstorcs** will abort, rather than overwrite those files if they do exist.

SEE ALSO

ci(1), co(1), rcs(1)

NOTES

Copyright 1983 by Kenneth L. Greer

DIAGNOSTICS

All diagnostics are written to *stderr*.

Non-zero exit status on error.

BUGS

sccstorcs does not preserve all SCCS options specified in the SCCS file. Most notably, it does not preserve removed deltas, MR numbers, and cutoff points.

NAME

sdiff – side-by-side difference program

SYNOPSIS

sdiff [options ...] file1 file2

DESCRIPTION

sdiff uses the output of **diff**(1) to produce a side-by-side listing of two files indicating those lines that are different. Each line of the two files is printed with a blank gutter between them if the lines are identical, a < in the gutter if the line only exists in *file1*, a > in the gutter if the line only exists in *file2*, and a | for lines that are different.

Following is a sample of the output **sdiff** produces:

```
x       |       y
a               a
b       <
c       <
d               d
        >       c
```

The following options exist:

–w *n* Use the next argument, *n*, as the width of the output line. The default line length is 130 characters.

–l Print only the left side of any lines that are identical.

–s Do not print identical lines.

–o *output* Use the next argument, *output*, as the name of a third file that is created as a user-controlled merging of *file1* and *file2*. Identical lines of *file1* and *file2* are copied to *output*. Sets of differences, as produced by **diff**(1), are printed; where a set of differences share a common gutter character. After printing each set of differences, **sdiff** prompts the user with a % and waits for the user to type one of the following commands:

l	Append the left column to the output file
r	Append the right column to the output file
s	Turn on silent mode; do not print identical lines
v	Turn off silent mode
e l	Call the editor with the left column
e r	Call the editor with the right column
e b	Call the editor with the concatenation of left and right
e	Call the editor with a zero-length file
q	Exit from the program

On exit from the editor, the resulting file is concatenated on the end of the *output* file.

SEE ALSO

diff(1), ed(1)

NAME

sed – stream editor

SYNOPSIS

sed [**–n**] [**–e** script] [**–f** sfile] [files]

DESCRIPTION

sed copies the named *files* (standard input default) to the standard output, edited according to a script of commands. The **–f** option causes the script to be taken from file *sfile*; these options accumulate. If there is just one **–e** option and no **–f** options, the flag **–e** can be omitted. The **–n** option suppresses the default output. A script consists of editing commands, one per line, of the following form:

[address [, address]] function [arguments]

In normal operation, **sed** cyclically copies a line of input into a pattern space (unless there is something left after a **D** command), applies in sequence all commands whose addresses select that pattern space, and at the end of the script copies the pattern space to the standard output (except under **–n**) and deletes the pattern space.

Some of the commands use a hold space to save all or part of the pattern space for subsequent retrieval.

An *address* is either a decimal number that counts input lines cumulatively across files, a **$** that addresses the last line of input, or a context address, i.e., a */regular expression/* in the style of **ed**(1), modified as follows:

- In a context address, the construction *\?regular expression?* (where *?* is any character) is identical to */regular expression/*. Note that in the context address **\xabc\xdefx**, the second **x** stands for itself, so that the regular expression is **abcxdef**.
- The escape sequence **\n** matches a newline embedded in the pattern space.
- A period (.) matches any character except the terminal newline of the pattern space.
- A command line with no addresses selects every pattern space.
- A command line with one address selects each pattern space that matches the address.
- A command line with two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line is selected.) Thereafter the process is repeated, looking again for the first address.

Editing commands can be applied only to non-selected pattern spaces by use of the negation function **!** (below).

In the following list of functions, the maximum number of permissible addresses for each function is indicated in parentheses.

The *text* argument consists of one or more lines, all but the last of which end with **** to hide the newline. Backslashes in text are treated like backslashes in

the replacement string of an **s** command, and may be used to protect initial blanks and tabs against the stripping that is done on every script line. The *rfile* or *wfile* argument must terminate the command line and must be preceded by exactly one blank. Each *wfile* is created before processing begins. There can be at most 10 distinct *wfile* arguments.

(1)**a**
text
: Append. Place *text* on the output before reading the next input line.

(2)**b** *label*
: Branch to the **:** command bearing the *label*. If *label* is empty, branch to the end of the script.

(2)**c**
text
: Change. Delete the pattern space. With 0 or 1 address or at the end of a 2-address range, place *text* on the output. Start the next cycle.

(2)**d**
: Delete the pattern space. Start the next cycle.

(2)**D**
: Delete the initial segment of the pattern space through the first newline. Start the next cycle.

(2)**g**
: Replace the contents of the pattern space by the contents of the hold space.

(2)**G**
: Append the contents of the hold space to the pattern space.

(2)**h**
: Replace the contents of the hold space by the contents of the pattern space.

(2)**H**
: Append the contents of the pattern space to the hold space.

(1)**i**
text
: Insert. Place *text* on the standard output.

(2)**l**
: List the pattern space on the standard output in an unambiguous form. Non-printable characters are displayed in octal notation and long lines are folded.

(2)**n**
: Copy the pattern space to the standard output. Replace the pattern space with the next line of input.

(2)**N**
: Append the next line of input to the pattern space with an embedded newline. (The current line number changes.)

(2)**p**
: Print. Copy the pattern space to the standard output.

(2)**P**
: Copy the initial segment of the pattern space through the first newline to the standard output.

(1)**q**
: Quit. Branch to the end of the script. Do not start a new cycle.

(2)**r** *rfile*
: Read the contents of *rfile*. Place them on the output before reading the next input line.

(2)**s**/*regular expression*/*replacement*/*flags*
: Substitute the *replacement* string for instances of the *regular expression* in the pattern space. Any character may be used instead of /. For a fuller description, see **ed**(1). *flags* can be zero or more of the following:

n n= 1 - 512. Substitute for just the n-th occurrence of the *regular expression.*

g Global. Substitute for all nonoverlapping instances of the *regular expression* rather than just the first one.

p Print the pattern space if a replacement was made.

w *wfile* Write. Append the pattern space to *wfile* if a replacement was made.

(2)**t** *label* Test. Branch to the **:** command bearing the *label* if any substitutions have been made since the most recent reading of an input line or execution of a **t**. If *label* is empty, branch to the end of the script.

(2)**w** *wfile* Write. Append the pattern space to *wfile*.

(2)**x** Exchange the contents of the pattern and hold spaces.

(2)**y**/*string1*/*string2*/
Transform. Replace all occurrences of characters in *string1* with the corresponding character in *string2*. The lengths of *string1* and *string2* must be equal.

(2)**!** *function* Don't. Apply the *function* (or group, if *function* is { } only to lines *not* selected by the address(es).

(0)**:** *label* This command does nothing; it bears a *label* for **b** and **t** commands to branch to.

(1)**=** Place the current line number on the standard output as a line.

(2){ Execute the following commands through a matching } only when the pattern space is selected.

(0) An empty command is ignored.

(0)**#** If a # appears as the first character on the first line of a script file, then that entire line is treated as a comment, with one exception. If the character after the # is an 'n', then the default output will be suppressed. The rest of the line after #n is also ignored. A script file must contain at least one non-comment line.

SEE ALSO

awk(1), ed(1), grep(1)

NAME

set, unset – set and unset environment variables in core or in files

SYNOPSIS

set –<l | filename | e variable=value> . . . | set –<l | f filename | e> variable

unset –<l | ffilename variable> . . .

DESCRIPTION

The **set** built-in can be used to set variables in the environment or to add variables to environment-like files. The **unset** built-in removes these variables. There are two built-in environments; a local one and the operating system environment which passes variables between processes. These are accessed by the **–l** and **–e** options, respectively. When expanding variables, the Interpreter checks the local environment first, then the operating system environment. If you use a *filename* with the **–f** option, you must include that *filename* when you are expanding variables (e.g. ${*(filename)VARIABLE*}).

If a *variable* name is given without equating it to a value, **set** expects the value to be on *stdin*.

EXAMPLES

- Storing a selection made in a menu:

```
.
.
.
name=Selection 2
action='set -l SELECTION=2'close
.
.
.
```

SEE ALSO

env(1), sh(1)

WARNINGS

Note that at least one of the allowed switches *must* be used.

Note that a switch must be used for each variable being set or unset.

Note that there is no space between the **–f** option and the filename.

Operating system environment variables (those set using the **–e**) can be set only for the current FMLI process and the processes it calls.

When the **–f** option is specified, other users of the Interpreter on the same machine can expand these variables unless the *filename* is unique to the process.

NAME

setcolor – redefine or create a color

SYNOPSIS

setcolor color red_level green_level blue_level

DESCRIPTION

The **setcolor** command takes four arguments; a string naming the color, and three integers defining the intensity of the red, green, and blue components of the color, respectively. If you are redefining an existing color, you must use its current name (the default colors are black, blue, green, cyan, red, magenta, yellow, and white). Intensities must be in the range of 0 to 1000. The function returns the color's name string.

EXAMPLES

```
'setcolor blue 100 24 300'
```

NAME

setmnt – establish mount table

SYNOPSIS

/etc/setmnt

DESCRIPTION

setmnt creates the */etc/mnttab* table, which is needed for both the **mount**(1M) and **umount** commands. **setmnt** reads standard input and creates a *mnttab* entry for each line. Input lines have this format:

filesys node

filesys is the name of the filesystem's special file (for example, */dev/dsk/zd0s0*) and *node* is the root name of that filesystem. Thus *filesys* and *node* become the first two items in the mount table entry.

FILES

/etc/mnttab

SEE ALSO

mount(1M)

BUGS

Problems may occur if *filesys* or *node* are longer than 32 characters.

setmnt silently enforces an upper limit on the maximum number of *mnttab* entries.

NAME

setpgrp – run command in its own process group

SYNOPSIS

setpgrp commands [args] ...

DESCRIPTION

Sets the process group ID of command to the process ID of command. This sets things so that command runs in its own process group.

SEE ALSO

setpgrp(2)

NAME

setup – initialize system for first user

SYNOPSIS

setup

DESCRIPTION

The **setup** command, which is also accessible as a login by the same name, allows the first user to be established as the *owner* of the machine.

The user is permitted to add the first logins to the system, usually starting with his or her own.

The user can then protect the system from unauthorized modification of the machine configuration and software by giving passwords to the administrative and maintenance functions. Normally, the first user of the machine enters this command through the setup login, which initially has no password, and then gives passwords to the various functions in the system. If the user does not assign a password to a function, it can be exercised by anyone.

The user can then give passwords to system logins such as **root** and **bin** (provided they do not already have passwords). Once given a password, each login can be changed only by that login or **root**.

The user can then set the date, time and time zone of the machine.

The user can then set the node name of the machine.

SEE ALSO

passwd(1)

DIAGNOSTICS

The **passwd**(1) command complains if the password provided does not meet its standards.

WARNINGS

If the setup login is not under password control, anyone can put passwords on the other functions.

NAME

sh, rsh – shell, the standard/restricted command programming language

SYNOPSIS

sh [**-acefhiknrstuvx**] [args]
rsh [**-acefhiknrstuvx**] [args]

DESCRIPTION

sh is a command programming language that executes commands read from a terminal or a file. **rsh** is a restricted version of the standard command interpreter **sh**; it is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. See **Invocation** below for the meaning of arguments to the shell.

Definitions

A blank is a tab or a space. A name is a sequence of letters, digits, or underscores beginning with a letter or underscore. A parameter is a name, a digit, or any of the characters *, @, #, ?, –, $, and !.

Commands

A simple-command is a sequence of non-blank words separated by blanks. The first word specifies the name of the command to be executed. Except as specified below, the remaining words are passed as arguments to the invoked command. The command name is passed as argument 0 (see **exec**(2)). The value of a simple-command is its exit status if it terminates normally, or (octal) 200+*status* if it terminates abnormally (see **signal**(2) for a list of status values).

A pipeline is a sequence of one or more commands separated by |. The standard output of each command but the last is connected by a **pipe**(2) to the standard input of the next command. Each command is run as a separate process; the shell waits for the last command to terminate. The exit status of a pipeline is the exit status of the last command.

A list is a sequence of one or more pipelines separated by ;, &, &&, or | |, and optionally terminated by ; or &. Of these four symbols, ; and & have equal precedence, which is lower than that of && and | |. The symbols && and | | also have equal precedence. A semicolon (;) causes sequential execution of the preceding pipeline; an ampersand (&) causes asynchronous execution of the preceding pipeline (i.e., the shell does not wait for that pipeline to finish). The symbol && (| |) causes the list following it to be executed only if the preceding pipeline returns a zero (non-zero) exit status. An arbitrary number of newlines (instead of semicolons) may appear in a list to delimit commands.

A command is either a simple-command or one of the following. Unless otherwise stated, the value returned by a command is that of the last simple-command executed in the command.

for *name* [**in** *word* ...] **do** *list* **done**

Each time a **for** command is executed, *name* is set to the next *word* taken from the **in** *word* list. If **in** *word* ... is omitted, then the **for** command executes the **do** *list* once for each positional parameter that is set (see **Parameter Substitution** below). Execution ends when there are no more words in the list.

case *word* **in** [*pattern* [| *pattern*] ...) *list* ;;] ... **esac**
A **case** command executes the *list* associated with the first *pattern* that matches *word*. The form of the patterns is the same as that used for filename generation (see **Filename Generation**) except that a slash, a leading dot, or a dot immediately following a slash need not be matched explicitly.

if *list* **then** *list* [**elif** *list* **then** *list*] ... [**else** *list*] **fi**
The *list* following **if** is executed and, if it returns a zero exit status, the *list* following the first **then** is executed. Otherwise, the *list* following **elif** is executed and, if its value is zero, the *list* following the next **then** is executed. Failing that, the **else** *list* is executed. If no **else** *list* or **then** *list* is executed, then the **if** command returns a zero exit status.

while *list* **do** *list* **done**
A **while** command repeatedly executes the **while** *list* and, if the exit status of the last command in the list is zero, executes the **do** *list*; otherwise the loop terminates. If no commands in the **do** *list* are executed, then the **while** command returns a zero exit status; **until** may be used in place of **while** to negate the loop termination test.

(*list*) Execute *list* in a sub-shell.

{*list*;} *list* is executed in the current (that is, parent) shell.

name () {*list*;}
Define a function which is referenced by *name*. The body of the function is the *list* of commands between { and }. Execution of functions is described below (see **Execution**).

The following words are only recognized as the first word of a command and when not quoted:

if then else elif fi case esac for while until do done { }

Comments

A word beginning with # causes that word and all the following characters up to a newline to be ignored.

Command Substitution

The shell reads commands from the string between two grave accents (``) and the standard output from these commands may be used as all or part of a word. Trailing newlines from the standard output are removed.

No interpretation is done on the string before the string is read, except to remove backslashes (\) used to escape other characters. Backslashes may be used to escape a grave accent (`) or another backslash (\) and are removed before the command string is read. Escaping grave accents allows nested command substitution. If the command substitution lies within a pair of double quotes (" ...` ...` ... "), a backslash used to escape a double quote (\") will be removed; otherwise, it will be left intact.

If a backslash is used to escape a newline character (\newline), both the backslash and the newline are removed (see the later section on **Quoting**). In addition, backslashes used to escape dollar signs (\$) are removed. Since no interpretation is done on the command string before it is read, inserting a

backslash to escape a dollar sign has no effect. Backslashes that precede characters other than backslash (\), grave accent (`), double quotes ("), newline, and dollar sign ($) are left intact when the command string is read.

Parameter Substitution

The character **$** is used to introduce substitutable parameters. There are two types of parameters, positional and keyword. If parameter is a digit, it is a positional parameter. Positional parameters may be assigned values by **set**. Keyword parameters (also known as variables) may be assigned values by writing:

name=*value* [*name*=*value*] ...

Pattern-matching is not performed on *value*. There cannot be a function and a variable with the same *name*.

${*parameter***}**

The value, if any, of the parameter is substituted. The braces are required only when *parameter* is followed by a letter, digit, or underscore that is not to be interpreted as part of its name. If *parameter* is * or @, all positional parameters, starting with **$1**, are substituted (separated by spaces). Parameter **$0** is set from argument zero when the shell is invoked.

${*parameter***:-***word***}**

If *parameter* is set and is non-null, substitute its value; otherwise substitute *word*.

${*parameter***:=***word***}**

If *parameter* is not set or is null set it to *word*; the value of the parameter is substituted. Positional parameters may not be assigned to in this way.

${*parameter***:?***word***}**

If *parameter* is set and is non-null, substitute its value; otherwise, print *word* and exit from the shell. If *word* is omitted, the message "parameter null or not set" is printed.

${*parameter***:+***word***}**

If *parameter* is set and is non-null, substitute *word*; otherwise substitute nothing.

In the above, *word* is not evaluated unless it is to be used as the substituted string. For example, in the following command **pwd** is executed only if **d** is not set or is null:

echo ${d:-`pwd`}

If the colon (:) is omitted from the above expressions, the shell checks only whether *parameter* is set.

The following parameters are automatically set by the shell:

The number of positional parameters in decimal.

- Flags supplied to the shell on invocation or by the **set** command.

? The decimal value returned by the last synchronously executed command.

$ The process number of this shell.

! The process number of the last background command invoked.

The following parameters are used by the shell:

HOME The default argument (home directory) for the **cd** command.

PATH The search path for commands (see **Execution** below). The user may not change **PATH** if executing under **rsh**.

CDPATH The search path for the **cd** command.

MAIL If this parameter is set to the name of a mail file and the **MAILPATH** parameter is not set, the shell informs the user of the arrival of mail in the specified file.

MAILCHECK This parameter specifies how often (in seconds) the shell checks for the arrival of mail in the files specified by the **MAILPATH** or **MAIL** parameters. The default value is 600 seconds (10 minutes). If set to 0, the shell checks before each prompt.

MAILPATH A colon (:) separated list of filenames. If this parameter is set, the shell informs the user of the arrival of mail in any of the specified files. Each filename can be followed by % and a message that will print when the modification time changes. The default message is *you have mail*.

PS1 Primary prompt string, by default "$ ".

PS2 Secondary prompt string, by default "> ".

IFS Internal field separators, normally space, tab, and newline.

SHACCT If this parameter is set to the name of a file writable by the user, the shell will write an accounting record in the file for each shell procedure executed.

SHELL When the shell is invoked, it scans the environment (see **Environment**) for this name. If it is found and **rsh** is the filename part of its value, the shell becomes a restricted shell.

The shell gives default values to **PATH**, **PS1**, **PS2**, **MAILCHECK** and **IFS**. **HOME** and **MAIL** are set by **login**(1).

Blank Interpretation

After parameter and command substitution, the results of substitution are scanned for internal field separator characters (those found in **IFS**) and split into distinct arguments where such characters are found. Explicit null arguments ("" or ´´) are retained. Implicit null arguments (those resulting from parameters that have no values) are removed.

Input/Output

A command's input and output may be redirected using a special notation interpreted by the shell. The following may appear anywhere in a simple-command or may precede or follow a command and are not passed on as arguments to the invoked command. Note that parameter and command substitution occurs before *word* or *digit* is used.

<word Use file *word* as standard input (file descriptor 0).

>word Use file *word* as standard output (file descriptor 1). If the file does not exist it is created. Otherwise, it is truncated to zero length.

>>word Use file *word* as standard output. If the file exists, output is appended to it (by first seeking to the end-of-file). Otherwise, the file is created.

<<[-]word After parameter and command substitution is done on *word*, the shell input is read up to the first line that literally matches the resulting *word*, or to an end-of-file. If, however, **-** is appended to **<<**, the following occurs:

1) Leading tabs are stripped from *word* before the shell input is read (but after parameter and command substitution is done on *word*).
2) Leading tabs are stripped from the shell input as it is read and before each line is compared with *word*.
3) Shell input is read up to the first line that literally matches the resulting *word*, or to an end-of-file.

If any character of *word* is quoted (see **Quoting**, later), no additional processing is done to the shell input. If no characters of *word* are quoted, the following takes place:

1) Parameter and command substitution occurs.
2) (Escaped) **\newline** is ignored.
3) **** must be used to quote the characters ****, **$**, and **`**.

The resulting document becomes the standard input.

<&digit Use the file associated with file descriptor *digit* as standard input. Similarly for the standard output using **>&digit**.

<&- The standard input is closed. Similarly for the standard output using **>&-**.

If any of the above is preceded by a digit, the file descriptor which will be associated with the file is that specified by the digit (instead of the default 0 or 1). For example, the following line associates file descriptor 2 with the file currently associated with file descriptor 1.

```
... 2>&1
```

The order in which redirections are specified is significant. The shell evaluates redirections left-to-right. For example, the following line first associates file descriptor 1 with file *xxx*. It associates file descriptor 2 with the file associated with file descriptor 1 (i.e., *xxx*).

```
... 1>xxx 2>&1
```

If the order of redirections were reversed, file descriptor 2 would be associated with the terminal (assuming file descriptor 1 had been) and file descriptor 1 would be associated with file *xxx*.

Using the terminology introduced under **Commands** on the first page, if a command is composed of several simple commands, redirection will be evaluated for the entire command before it is evaluated for each simple

command. That is, the shell evaluates redirection for the entire list, then each pipeline within the list, then each command within each pipeline, then each list within each command.

If a command is followed by &, the default standard input for the command is the empty file */dev/null*. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.

Redirection of output is not allowed in the restricted shell.

Filename Generation

Before a command is executed, each command *word* is scanned for the characters *, ?, and [. If one of these characters appears, the word is regarded as a *pattern*. The word is replaced with alphabetically sorted filenames that match the pattern. If no filename is found that matches the pattern, the word is left unchanged. The character . at the start of a filename or immediately following a /, as well as the character / itself, must be matched explicitly.

* Matches any string, including the null string.

? Matches any single character.

[...] Matches any one of the enclosed characters. A pair of characters separated by – matches any character lexically between the pair, inclusive. If the first character following the opening [is a exclamation mark (!), any character not enclosed is matched.

Quoting

The following characters have a special meaning to the shell and cause termination of a word unless quoted:

; & ( ) | ^ < > newline space tab

A character may be quoted (i.e., made to stand for itself) by preceding it with a backslash (\) or inserting it between a pair of quote marks (´´ or " "). During processing, the shell may quote certain characters to prevent them from taking on a special meaning. Backslashes used to quote a single character are removed from the word before the command is executed. The pair \newline is removed from a word before command and parameter substitution.

All characters enclosed between a pair of single quote marks (´´), except a single quote, are quoted by the shell. Backslash has no special meaning inside a pair of single quotes. A single quote may be quoted inside a pair of double quote marks (for example, "´").

Inside a pair of double quotes (" "), parameter and command substitution occurs and the shell quotes the results to avoid blank interpretation and filename generation. If **$*** is within a pair of double quotes, the positional parameters are substituted and quoted, separated by quoted spaces ("$1 $2 ..."). However, if **$@** is within a pair of double quotes, the positional parameters are substituted and quoted, separated by unquoted spaces ("$1" "$2" ...). \ quotes the characters \, `, ", and $. The pair \newline is removed before parameter and command substitution. If a backslash precedes characters other than \, `, ", $, and newline, then the backslash itself is quoted by the shell.

Prompting

When used interactively, the shell prompts with the value of PS1 before reading a command. If at any time a newline is typed and further input is needed to complete a command, the secondary prompt (i.e., the value of **PS2**) is issued.

Environment

The environment (see **environ**(5)) is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a parameter for each name found, giving it the corresponding value. If the user modifies the value of any of these parameters or creates new parameters, none of these affects the environment unless the **export** command is used to bind the shell's parameter to the environment (see also **set –a**). A parameter may be removed from the environment with the **unset** command. The environment seen by any executed command is thus composed of any unmodified name-value pairs originally inherited by the shell, minus any pairs removed by **unset**, plus any modifications or additions, all of which must be noted in **export** commands.

The environment for any simple-command may be augmented by prefixing it with one or more assignments to parameters. Thus, the following lines are equivalent (as far as the execution of *cmd* is concerned).

```
TERM=450 cmd
and
(export TERM; TERM=450; cmd)
```

If the **–k** flag is set, *all* keyword arguments are placed in the environment, even if they occur after the command name. The following first prints **a=b c** and **c**:

```
echo a=b c
set –k
echo a=b c
```

Signals

The INTERRUPT and QUIT signals for an invoked command are ignored if the command is followed by **&**; otherwise signals have the values inherited by the shell from its parent, with the exception of signal 11 (see the **trap** command).

Execution

Each time a command is executed, the above substitutions are carried out. If the command name matches one of the Special Commands listed below, it is executed in the shell process. If the command name does not match a Special Command but matches the name of a defined function, the function is executed in the shell process (note how this differs from the execution of shell procedures). The positional parameters **$1**, **$2**, ... are set to the arguments of the function. If the command name matches neither a Special Command nor the name of a defined function, a new process is created and an attempt is made to execute the command via **exec**(2).

The shell parameter **PATH** defines the search path for the directory containing the command. Alternative directory names are separated by a colon (:). The default path is *:/bin:/usr/bin* (specifying the current directory, */bin*, and */usr/bin*, in that order). The current directory is specified by a null pathname, which can appear immediately after the equal sign, between two colon

delimiters anywhere in the path list, or at the end of the path list. If the command name contains a /, the search path is not used; such commands will not be executed by the restricted shell. Otherwise, each directory in the path is searched for an executable file. If the file has execute permission but is not an *a.out* file, it is assumed to be a file containing shell commands. A sub-shell is spawned to read it. A parenthesized command is also executed in a sub-shell.

The location in the search path where a command was found is remembered by the shell (to avoid unnecessary **execs** later). If the command was found in a relative directory, its location must be re-determined whenever the current directory changes. The shell forgets all remembered locations whenever the **PATH** variable is changed or the **hash –r** command is executed (see below).

Special Commands

Input/output redirection is now permitted for these commands. File descriptor 1 is the default output location.

: No effect; the command does nothing. A zero exit code is returned.

. *file* Read and execute commands from *file* and return. The search path specified by **PATH** is used to find the directory containing *file*.

break [*n*]
Exit from the enclosing **for** or **while** loop, if any. If *n* is specified break *n* levels.

continue [*n*]
Resume the next iteration of the enclosing **for** or **while** loop. If *n* is specified resume at the *n*-th enclosing loop.

cd [*arg*]
Change the current directory to *arg*. The shell parameter **HOME** is the default *arg*. The shell parameter **CDPATH** defines the search path for the directory containing *arg*. Alternative directory names are separated by a colon. The default path is *<null>* (specifying the current directory). Note that the current directory is specified by a null pathname, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If *arg* begins with a / the search path is not used. Otherwise, each directory in the path is searched for *arg*. **rsh** cannot execute the **cd** command.

echo [*arg* ...]
Echo arguments. See **echo**(1) for usage and description.

eval [*arg* ...]
The arguments are read as input to the shell and the resulting command(s) executed.

exec [*arg* ...]
The command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments may appear and, if no other arguments are given, cause the shell input/output to be modified.

exit [*n*]
Causes a shell to exit with the exit status specified by *n*. If *n* is omitted the exit status is that of the last command executed (an end-of-file

will also cause the shell to exit).

export [*name* ...]
The given *name*s are marked for automatic export to the *environment* of subsequently-executed commands. If no arguments are given, variable names that have been marked for export during the current shell's execution are listed. (Variable names exported from a parent shell are listed only if they have been exported again during the current shell's execution.) Function names are *not* exported.

getopts
Use in shell scripts to support command syntax standards (see **intro**(1)); it parses positional parameters and checks for legal options. See **getopts**(1) for usage and description.

hash [**–r**] [*name* ...]
For each *name*, the location in the search path of the command specified by *name* is determined and remembered by the shell. The **–r** option causes the shell to forget all remembered locations. If no arguments are given, information about remembered commands is presented. *Hits* is the number of times a command has been invoked by the shell process. *Cost* is a measure of the work required to locate a command in the search path. If a command is found in a "relative" directory in the search path, after changing to that directory, the stored location of that command is recalculated. Commands for which this will be done are indicated by an asterisk (*) adjacent to the *hits* information. *Cost* will be incremented when the recalculation is done.

newgrp [*arg* ...]
Equivalent to **exec newgrp** *arg* .... See **newgrp**(1) for usage and description.

pwd Print the current working directory. See **pwd**(1) for usage and description.

read [*name* ...]
One line is read from the standard input and, using the internal field separator **IFS** (normally space or tab) to delimit word boundaries, the first word is assigned to the first *name*, the second word to the second *name*, etc., with leftover words assigned to the last *name*. Lines can be continued by using \newline. Characters other than newline can be quoted by preceding them with a backslash. These backslashes are removed before words are assigned to *names*, and no interpretation is done on the character that follows the backslash. The return code is 0 unless an end-of-file is encountered.

readonly [*name* ...]
The given *name*s are marked *readonly* and the values of these *name*s may not be changed by subsequent assignment. If no arguments are given, a list of all *readonly* names is printed.

return [*n*]
Causes a function to exit with the return value specified by *n*. If *n* is omitted, the return status is that of the last command executed.

set [**—aefhkntuvx** [*arg* ...]]

–a Mark variables which are modified or created for export.

–e Exit immediately if a command exits with a non-zero exit status.

–f Disable filename generation.

–h Locate and remember function commands as functions are defined (function commands are normally located when the function is executed).

–k All keyword arguments are placed in the environment for a command, not just those that precede the command name.

–n Read commands but do not execute them.

–t Exit after reading and executing one command.

–u Treat unset variables as an error when substituting.

–v Print shell input lines as they are read.

–x Print commands and their arguments as they are executed.

–– Do not change any of the flags; useful in setting **$1** to –.

Using rather than – causes these flags to be turned off. These flags can also be used upon invocation of the shell. The current set of flags may be found in **$–**. The remaining arguments are positional parameters and are assigned, in order, to **$1**, **$2**, ... If no arguments are given the values of all names are printed.

shift [*n*]
The positional parameters from **$n+1**... are renamed **$1**.... If *n* is not given, it is assumed to be 1.

test Evaluate conditional expressions. See **test**(1) for usage and description.

times Print the accumulated user and system times for processes run from the shell.

trap [*arg*][*n*]...
The command *arg* is to be read and executed when the shell receives signal(s) *n*. (Note that *arg* is scanned once when the trap is set and once when the trap is taken.) Trap commands are executed in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective. An attempt to trap on signal 11 (memory fault) produces an error. If *arg* is absent, all trap(s) *n* are reset to their original values. If *arg* is the null string this signal is ignored by the shell and by the commands it invokes. If *n* is 0, the command *arg* is executed on exit from the shell. The **trap** command with no arguments prints a list of commands associated with each signal number.

type [*name* ...]
For each *name*, indicate how it would be interpreted if used as a command name.

ulimit [*n*]
Impose a size limit of *n* blocks on files written by the shell and its

child processes (files of any size may be read). If *n* is omitted, the current limit is printed. You may lower your own ulimit, but only a superuser (see **su**(1M)) can raise a ulimit.

umask [*nnn*]
The user file-creation mask is set to *nnn* (see **umask**(1)). If *nnn* is omitted, the current value of the mask is printed.

unset [*name* ...]
For each *name*, remove the corresponding variable or function. The variables **PATH, PS1, PS2, MAILCHECK** and **IFS** cannot be unset.

wait [*n*]
Wait for your background process whose process ID is *n* and report its termination status. If *n* is omitted, all your shell's currently active background processes are waited for and the return code will be zero.

Invocation

If the shell is invoked through **exec**(2) and the first character of argument zero is –, commands are initially read from */etc/profile* and from *$HOME/.profile*, if such files exist. Thereafter, commands are read as described below, which is also the case when the shell is invoked as **/bin/sh**. The flags below are interpreted by the shell on invocation only. Note that unless the **–c** or **–s** flag is specified, the first argument is assumed to be the name of a file containing commands, and the remaining arguments are passed as positional parameters to that command file.

–c *string* If the **–c** flag is present, commands are read from *string*.

–s If the **–s** flag is present or if no arguments remain, commands are read from the standard input. Any remaining arguments specify the positional parameters. Shell output (except for Special Commands) is written to file descriptor 2.

–i If the **–i** flag is present or if the shell input and output are attached to a terminal, this shell is *interactive*. In this case TERMINATE is ignored (so that **kill 0** does not kill an interactive shell) and INTERRUPT is caught and ignored (so that **wait** is interruptible). In all cases, QUIT is ignored by the shell.

–r If the **–r** flag is present, the shell is a restricted shell.

The remaining flags and arguments are described under the **set** command.

rsh Only

rsh is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. The actions of **rsh** are identical to those of **sh**, except that the following are disallowed:

Changing directory (see **cd**(1))

Setting the value of **$PATH**

Specifying path or command names containing /

Redirecting output (> and >>)

The restrictions above are enforced after *.profile* is interpreted.

A restricted shell can be invoked in one of the following ways:

(1) **rsh** is the filename part of the last entry in the */etc/passwd* file (see **passwd**(4)).

(2) The environment variable **SHELL** exists and **rsh** is the filename part of its value.

(3) The shell is invoked and **rsh** is the filename part of argument 0.

(4) The shell is invoked with the **–r** option.

When a command to be executed is found to be a shell procedure, **rsh** invokes **sh** to execute it. Thus, it is possible to provide to the end-user shell procedures that have access to the full power of the standard shell, while imposing a limited menu of commands. This scheme assumes that the end-user does not have write and execute permissions in the same directory.

The net effect of these rules is that the writer of the *.profile* (see **profile**(4)) has complete control over user actions by performing guaranteed setup actions and leaving the user in an appropriate directory (probably not the login directory).

The system administrator often sets up a directory of commands (such as */usr/rbin*) that can be safely invoked by a restricted shell. Some systems also provide a restricted editor, **red**.

PARALLEL SUPPORT

sh manages the creation of command lists terminated with an ampersand in a manner similar to parallel **make**(1). The environment variable **PARALLEL** is used to determine the number of commands lists to be run concurrently. if **PARALLEL** is not set, **sh** will always attempt to run the command list.

FILES

/etc/profile
$HOME/.profile
/tmp/sh*
/dev/null

SEE ALSO

cd(1), echo(1), env(1), getopts(1), intro(1), login(1), newgrp(1), pwd(1), test(1), umask(1), wait(1), dup(2), exec(2), fork(2), pipe(2), signal(2), ulimit(2), profile(4)

DIAGNOSTICS

Errors detected by the shell, such as syntax errors, cause the shell to return a non-zero exit status. If the shell is being used non-interactively execution of the shell file is abandoned. Otherwise, the shell returns the exit status of the last command executed (see also the **exit** command above).

NOTES

Words used for filenames in input/output redirection are not interpreted for filename generation (see **Filename Generation**). For example, **cat file1 >a*** will create a file named **a***.

Because commands in pipelines are run as separate processes, variables set in a pipeline have no effect on the parent shell.

If you get the error message *cannot fork, too many processes*, try using the **wait**(1) command to clean up your background processes. If this doesn't help,

the system process table is probably full or you have too many active foreground processes. (There is a limit to the number of process ids associated with your login, and to the number the system can keep track of.)

BUGS

If a command is executed and a command with the same name is installed in a directory in the search path before the directory where the original command was found, the shell will continue to **exec** the original command. Use the **hash** command to correct this situation.

If you move the current directory or one above it, **pwd** may not give the correct response. Use the **cd** command with a full pathname to correct this situation.

Not all the processes of a 3- or more-stage pipeline are children of the shell, and thus cannot be waited for.

For **wait n**, if *n* is not an active process id, all your shell's currently active background processes are waited for and the return code will be zero.

NAME

shell – run a command using shell

SYNOPSIS

shell command [command ...]

DESCRIPTION

The **shell** built-in takes each of its arguments, puts them together separated by a space, and passes this command to the operating system shell (**$SHELL** if set, otherwise **/bin/sh**).

EXAMPLES

Since the Interpreter doesn't support background processing, the **shell** built-in can be used instead.

```
'shell "build prog &"'
```

Using **test**(1) can be useful. This example tests to see if field 2 of a form is a file.

```
valid='shell test -f $F2'
```

SEE ALSO

sh(1), test(1)

WARNINGS

The arguments are concatenated using spaces, which may or may not do what is expected. The variables set in local environments are not expanded by the shell because local means local to the current process.

NAME

showcfg – show the configuration of the machine

SYNOPSIS

/etc/showcfg [**–s** | **–d**]

DESCRIPTION

showcfg displays the configuration of the system in a manner similar to the **rc** command of the SCED power-up monitor and the **config** command of the SSM power-up monitor.

The **–s** flag selects an alternate one-line format that gives the quantity of each type of board.

The **–d** flag produces a dump of relevant parts of the system's configuration description table. The data displayed includes information about the memory available, the boot flags, the boot device, console tty control characters, and the current system bus mode.

FILES

/dev/kmem

SEE ALSO

dmesg(1M)
/usr/adm/messages

BUGS

Boards on the MULTIBUS or VMEBUS are not currently displayed.

NAME

shutdown – shut down system, change system state

SYNOPSIS

/etc/shutdown [**-y**] [**-g***grace_period*] [**-i***init_state*]

DESCRIPTION

This command is executed by the superuser to change the state of the machine. By default, it brings the system to a state where only the console has access to the operating system. This state is traditionally called *single-user*.

The command sends a warning message and a final message before it starts actual shutdown activities. By default, the command asks for confirmation before it starts shutting down daemons and killing processes. The options are used as follows:

-y Pre-answers the confirmation question so the command can be run without user intervention. A default of 60 seconds is allowed between the warning message and the final message. Another 60 seconds is allowed between the final message and the confirmation.

-g*grace_period*
Allows the superuser to change the number of seconds from the 60-second default.

-i*init_state*
Specifies the state that **init**(1M) is to be put in following the warnings, if any. By default, system state **s** is used (the same as states **1** and **S**).

Other recommended system state definitions are as follows:

state 0
Shut the machine down so it is safe to remove the power. Have the machine remove power if it can. The **/etc/rc0** procedure is called to do this work.

state 1, s, S
Bring the machine to the state traditionally called single-user. The **/etc/rc0** procedure is called to do this work. (Though **s** and **1** are both used to go to single-user state, **s** only kills processes spawned by **init** and does not unmount filesystems. State **1** unmounts everything except root and kills all user processes, except those that relate to the console.)

state 5
Stop the operating system and go to the firmware monitor.

state 6
Stop the operating system and reboot to the state defined by the *initdefault* entry in */etc/inittab*. Reboot occurs only if the AUTO button on the front of the machine is pushed (i.e. autobooting is enabled). If autobooting is not enabled, the operating system is stopped and goes to the firmware monitor (same as state 5).

SEE ALSO

init(1M), rc0(1M), rc2(1M), inittab(4)

NAME

sleep – suspend execution for an interval

SYNOPSIS

sleep time

DESCRIPTION

sleep suspends execution for *time* seconds. It is used to execute a command after a certain amount of time, as in the following example:

```
(sleep 105; command)&
```

It can also execute a command at specified times, as in the next example:

```
while true
do
        command
        sleep 37
done
```

SEE ALSO

alarm(2), sleep(3C)

NAME

sort – sort and/or merge files

SYNOPSIS

sort [**–cmu**] [**–o**output] [**–y**kmem] [**–z**recsz] [**–dfiMnr**] [**–btx**] [+pos1 [–pos2]] [files]

DESCRIPTION

sort sorts lines of all the named files together and writes the result on the standard output. The standard input is read if – is used as a filename or no input files are named.

Comparisons are based on one or more sort keys extracted from each line of input. By default, there is one sort key (the entire input line) and ordering is lexicographic by bytes in machine collating sequence.

The following options alter the default behavior:

–c Check that the input file is sorted according to the ordering rules; give no output unless the file is out of sort.

–m Merge only, the input files are already sorted.

–u Unique: suppress all but one in each set of lines having equal keys.

–o*output* The argument given is the name of an output file to use instead of the standard output. This file may be the same as one of the inputs. There may be optional blanks between **–o** and *output*.

–y*kmem* The amount of main memory used by the sort has a large impact on its performance. Sorting a small file in a large amount of memory is a waste. If this option is omitted, **sort** begins using a system default memory size, and continues to use more space as needed. If this option is presented with a value, *kmem*, **sort** will start using that number of kilobytes of memory, unless the administrative minimum or maximum is violated, in which case the corresponding extremum will be used. Thus, **–y0** is guaranteed to start with minimum memory. By convention, **–y** (with no argument) starts with maximum memory.

–z*recsz* The size of the longest line read is recorded in the sort phase so buffers can be allocated during the merge phase. If the sort phase is omitted via the **–c** or **–m** options, a popular system default size will be used. Lines longer than the buffer size will cause **sort** to terminate abnormally. Supplying the actual number of bytes in the longest line to be merged (or some larger value) will prevent abnormal termination.

The following options override the default ordering rules:

–d "Dictionary" order: only letters, digits, and blanks (spaces and tabs) are significant in comparisons.

–f Fold lowercase letters into uppercase.

–i Ignore non-printable characters.

–M Compare as months. The first three non-blank characters of the field are folded to uppercase and compared. For example, in English the sorting order is "JAN" < "FEB" < ... < "DEC". Invalid fields compare low to "JAN".

The **-M** option implies the **-b** option (see below).

-n An initial numeric string, consisting of optional blanks, optional minus sign, and zero or more digits with optional decimal point, is sorted by arithmetic value. The **-n** option implies the **-b** option (see below). Note that the **-b** option is effective only when restricted sort key specifications are in effect.

-r Reverse the sense of comparisons.

When ordering options appear before restricted sort key specifications, the requested ordering rules are applied globally to all sort keys. When attached to a specific sort key (described below), the specified ordering options override all global ordering options for that key.

The notation *+pos1 -pos2* restricts a sort key to one beginning at *pos1* and ending just before *pos2*. The characters at position *pos1* and just before *pos2* are included in the sort key (provided that *pos2* does not precede *pos1*). A missing *-pos2* means the end of the line.

Specifying *pos1* and *pos2* involves the notion of a field, a minimal sequence of characters followed by a field separator or a newline. By default, the first blank (space or tab) of a sequence of blanks acts as the field separator. All blanks in a sequence of blanks are considered to be part of the next field. For example, all blanks at the beginning of a line are considered to be part of the first field. The treatment of field separators can be altered by using these options:

-b Ignore leading blanks when determining the starting and ending positions of a restricted sort key. If the **-b** option is specified before the first *+pos1* argument, it will be applied to all *+pos1* arguments. Otherwise, the **b** flag may be attached independently to each *+pos1* or *-pos2* argument (see below).

-t*x* Use *x* as the field separator character. *x* is not considered to be part of a field (although it may be included in a sort key). Each occurrence of *x* is significant (for example, *xx* delimits an empty field).

Pos1 and *pos2* each have the form *m.n* optionally followed by one or more of the flags **bdfinr**. A starting position specified by *+m.n* is interpreted to mean the *n*+1st character in the *m*+1st field. A missing *.n* means .0, indicating the first character of the *m*+1st field. If the **b** flag is in effect, *n* is counted from the first non-blank in the *m*+1st field; *+m*.0**b** refers to the first non-blank character in the *m*+1st field.

A last position specified by *-m.n* is interpreted to mean the *n*th character (including separators) after the last character of the *m*th field. A missing *.n* means .0, indicating the last character of the *m*th field. If the **b** flag is in effect, *n* is counted from the last leading blank in the *m*+1st field; *-m*.1**b** refers to the first non-blank in the *m*+1st field.

When there are multiple sort keys, later keys are compared only after all earlier keys compare equal. Lines that otherwise compare equal are ordered with all bytes significant.

EXAMPLES

Sort the contents of *infile* with the second field as the sort key:

sort +1 –2 infile

Sort, in reverse order, the contents of *infile1* and *infile2*, placing the output in *outfile* and using the first character of the second field as the sort key:

sort –r –o outfile +1.0 –1.2 infile1 infile2

Sort, in reverse order, the contents of *infile1* and *infile2* using the first non-blank character of the second field as the sort key:

sort –r +1.0b –1.1b infile1 infile2

Print the password file (**passwd**(4)) sorted by the numeric user ID (the third colon-separated field):

sort –t: +2n –3 /etc/passwd

Print the lines of the already sorted file *infile*, suppressing all but the first occurrence of lines having the same third field (the options **–um** with just one input file make the choice of a unique representative from a set of equal lines predictable):

sort –um +2 –3 infile

FILES

/usr/tmp/stm???

SEE ALSO

comm(1), join(1), uniq(1)

WARNINGS

Comments and exits with non-zero status for various trouble conditions (for example, when input lines are too long), and for disorder discovered under the **–c** option. When the last line of an input file is missing a newline character, **sort** appends one, prints a warning message, and continues.

sort does not guarantee preservation of relative line ordering on equal keys.

NAME

spell, hashmake, spellin, hashcheck – find spelling errors

SYNOPSIS

spell [**–v**] [**–b**] [**–x**] [**–l**] [+local_file] [files]

/usr/lib/spell/hashmake

/usr/lib/spell/spellin n

/usr/lib/spell/hashcheck spelling_list

DESCRIPTION

spell collects words from the named *files* and looks them up in a spelling list. Words that do not occur among or are not derivable (by applying certain inflections, prefixes, and/or suffixes) from words in the spelling list are printed on the standard output. If no *files* are named, words are collected from the standard input.

spell ignores most **tbl**(1) and **eqn**(1) constructions.

Under the **–v** option, all words not literally in the spelling list are printed and plausible derivations from the words in the spelling list are indicated.

Under the **–b** option, British spelling is checked. Besides preferring centre, colour, programme, speciality, travelled, etc., this option insists upon -ise in words like standardise, Fowler and the OED to the contrary notwithstanding.

Under the **–x** option, every plausible stem is printed with = for each word.

By default, **spell** (like **deroff**(1)) follows chains of included files (*.so* requests) unless the names of such included files begin with */usr/lib*. Under the **–l** option, **spell** will follow the chains of all included files.

Under the *+local_file* option, words found in *local_file* are removed from **spell**'s output. *Local_file* is the name of a user-provided file that contains a sorted list of words, one per line. With this option, the user can specify a set of words that are correct spellings (in addition to **spell**'s own spelling list) for each job.

The spelling list is based on many sources, and while more haphazard than an ordinary dictionary, is also more effective with respect to proper names and popular technical words. Coverage of the specialized vocabularies of biology, medicine, and chemistry is light.

Pertinent auxiliary files may be specified by name arguments, indicated below with their default settings (see **FILES**). Copies of all output are accumulated in the history file. The stop list filters out misspellings (for example, thier=thy–y+ier) that would otherwise pass.

Three routines help maintain and check the hash lists used by **spell**:

hashmake Reads a list of words from the standard input and writes the corresponding nine-digit hash code on the standard output.

spellin Reads *n* hash codes from the standard input and writes a compressed spelling list on the standard output.

hashcheck Reads a compressed *spelling_list* and recreates the nine-digit hash codes for all the words in it. **hashcheck** then writes these codes on the standard output.

FILES

D_SPELL=/usr/lib/spell/hlist[ab]	Hashed spelling lists, American & British
S_SPELL=/usr/lib/spell/hstop	Hashed stop list
H_SPELL=/usr/lib/spell/spellhist	History file
/usr/lib/spell/spellprog	Program

SEE ALSO

deroff(1), sed(1), sort(1), tee(1)

eqn(1), tbl(1) in the *DOCUMENTER'S WORKBENCH Technical Discussion and Reference Manual*

BUGS

The spelling list's coverage is uneven. New installations will probably wish to monitor the output for several months to gather local additions. Typically, these are kept in a separate local file that is added to the hashed *spelling_list* via **spellin**.

NAME

spline – interpolate smooth curve

SYNOPSIS

spline [options]

DESCRIPTION

spline takes pairs of numbers from the standard input as abscissas and ordinates of a function. It produces a similar set (which is approximately equally spaced and includes the input set) on the standard output. The cubic spline output has two continuous derivatives and sufficiently many points to look smooth when plotted, for example by **graph**(1G).

The following options are recognized, each as a separate argument:

-a Supply abscissas automatically (they are missing from the input); spacing is given by the next argument, or is assumed to be 1 if next argument is not a number.

-k The constant k used in the boundary value computation shown below is set by the next argument (default $k = 0$).

$$y_0'' = ky_1'', \quad y_n'' = ky_{n-1}''$$

-n Space output points so that approximately n intervals occur between the lower and upper x limits (default $n = 100$).

-p Make output periodic, i.e., match derivatives at ends. First and last input values should normally agree.

-x Next 1 (or 2) arguments are lower (and upper) x limits. Normally, these limits are calculated from the data. Automatic abscissas start at lower limit (default 0).

SEE ALSO

graph(1G)

DIAGNOSTICS

When data is not strictly monotone in x, **spline** reproduces the input without interpolating extra points.

BUGS

A limit of 1000 input points is enforced silently.

NAME

split – split a file into pieces

SYNOPSIS

split [*–n*] [file [name]]

DESCRIPTION

split reads *file* and writes it in *n*-line pieces (default 1000 lines) onto a set of output files. The name of the first output file is *name* with *aa* appended, and so on lexicographically, up to *zz* (a maximum of 676 files). If no output name is given, *x* is default.

If no input file is given or if – is given, then the standard input file is used.

SEE ALSO

bfs(1), csplit(1)

NAME

starter – information about the operating system for beginning users

SYNOPSIS

[**help**] **starter**

DESCRIPTION

The Help Facility **starter** command provides information to assist new users:

- Commands a new user should learn first
- Local environment information

The user can choose one of the above categories by entering its corresponding letter (given in the menu), or can exit to the shell by typing q (for "quit"). When a category is chosen, the user will receive one or more pages of information pertaining to it.

From any screen in the Help Facility, a user can execute a command via the shell by typing an exclamation mark (!) and the command to be executed. The screen will be redrawn if the command was entered at a first-level prompt. If entered at any other prompt level, only the prompt will be redrawn.

By default, the Help Facility scrolls the data that is presented to the user. If you prefer to have the screen clear before printing the data (non-scrolling), the shell variable **SCROLL** must be set to **no** and exported so it will become part of your environment. This is done by adding the following line to your *.profile* file (see **profile**(4)): "`export SCROLL ; SCROLL=no`". If you later decide that scrolling is desired, **SCROLL** must be set to **yes**.

Information on each of the Help Facility commands (**starter**, **locate**, **usage**, **glossary**, and **help**) is located on their respective manual pages.

SEE ALSO

csh(1), glossary(1), help(1), locate(1), sh(1), usage(1), term(5)

WARNINGS

If the shell variable **TERM** is not set in the user's *.profile* file, then **TERM** will default to the terminal value type 450 (a hard-copy terminal). For a list of valid terminal types, refer to **term**(5).

NAME

stat – statistical network useful with graphical commands

SYNOPSIS

node-name [options] [files]

DESCRIPTION

stat is a collection of command level functions (nodes) that can be interconnected using **sh**(1) to form a statistical network. The nodes reside in */usr/bin/graf* (see **graphics**(1G)). Data is passed through the network as sequences of numbers (vectors), where a number is of the following form and is evaluated in the usual way.

[sign](digits)(.digits)[e[sign]digits]

Brackets and parentheses surround fields. All fields are optional, but at least one of the fields surrounded by parentheses must be present. Any character input to a node that is not part of a number is taken as a delimiter.

stat nodes are divided into four classes:

Transformers	Map input vector elements into output vector elements
Summarizers	Calculate statistics of a vector
Translators	Convert among formats
Generators	Sources of definable vectors

Below is a list of synopses for **stat** nodes. Most nodes accept options indicated by a leading hyphen (–). In general, an option is specified by a character followed by a value, such as **c5**. This is interpreted as **c** := 5 (**c** is assigned 5). The following keys are used to designate the expected type of the value:

c	Characters
i	Integer
f	Floating point or integer
file	Filename
string	String of characters surrounded by quotes to include a shell argument delimiter.

Options without keys are flags. All nodes except *generators* accept files as input, hence it is not indicated in the synopses.

Transformers:

abs	[–*ci*] – Absolute value columns (similarly for –**c** options that follow)
af	[–*ci* **t v**] – Arithmetic function titled output, verbose
ceil	[–*ci*] – Round up to next integer
cusum	[–*ci*] – Cumulative sum
exp	[–*ci*] – Exponential
floor	[–*ci*] – Round down to next integer
gamma	[–*ci*] – Gamma

list [*–ci* **d***string*] – List vector elements
delimiter(s)

log [*–ci* **b***f*] – Logarithm
base

mod [*–ci* **m***f*] – Modulus
modulus

pair [*–ci* **F***file* **x***i*] – Pair elements
File containing base vector, **x** group size

power [*–ci* **p***f*] – Raise to a power
power

root [*–ci* **r***f*] – Take a root
root

round [*–ci* **p***i* **s***i*] – Round to nearest integer, .5 rounds to 1
places after decimal point, significant digits

siline [*–ci* **i***f* **n***i* **s***f*] – Generate a line given slope and intercept
intercept, number of positive integers, slope

sin [*–ci*] – Sine

subset [*–a**f* **b***f* **c***i* **F***file* **i***i* **l***f* **nl np p***f* **s***i* **t***i*] – Generate a subset
above, below, File with master vector, interval, leave, master contains element numbers to leave, master contains element numbers to pick, pick, start, terminate

Summarizers:

bucket [*–a**i* **c***i* **F***file* **h***f* **i***i* **l***f* **n***i*] – Break into buckets
average size, File containing bucket boundaries, high, interval, low, number
Input data should be sorted

cor [*–F**file*] – Correlation coefficient
File containing base vector

hilo [– **h l o ox oy**]– Find high and low values
high only, low only, option form, option form with **x** prepended, option form with **y** prepended

lreg [*–F**file* **i o s**] – Linear regression
File containing base vector, intercept only, option form for *siline*, slope only

mean [*–f**f* **n***i* **p***f*] – (Trimmed) arithmetic mean
fraction, number, percent

point [*–f**f* **n***i* **p***f* **s**] – Point from empirical cumulative density function
fraction, number, percent, sorted input

prod – Internal product

qsort [*–ci*] – Quick sort

rank – Vector rank

total – Sum total

var – Variance

Translators:

bar [**-a b f g** *ri* **w** *i* **x** *f* **xa y** *f* **ya yl** *f* **yh** *f*] – Build a bar chart
Suppress axes, bold, suppress frame, suppress grid, region, width in percent, x origin, suppress x-axis label, y origin, suppress y-axis label, y-axis lower bound, y-axis high bound
Data is rounded off to integers.

hist [**-a b f g** *ri* **x** *f* **xa y** *f* **ya yl** *f* **yh** *f*] – Build a histogram
Suppress axes, bold, suppress frame, suppress grid, region, x origin, suppress x-axis label, y origin, suppress y-axis label, y-axis lower bound, y-axis high bound

label [**-b c F** *file* **h p ri x xu y yr**] – Label the axis of a GPS file
bar chart input, retain case, label File, histogram input, plot input, rotation, x-axis, upper x-axis, y-axis, right y-axis

pie [**-b o p pn** *i* **pp** *i* **ri v x** *i* **y** *i*] – Build a pie chart
bold, values outside pie, value as percentage(:=100), value as percentage(:=i), draw percent of pie, region, no values, x origin, y origin
Unlike other nodes, input is lines of the form:

[< **i e f c** *c* >] value [label]
ignore (do not draw) slice, explode slice, fill slice, color slice *c*=(black, red, green, blue)

plot [**-a b c** *string* **d f F** *file* **g m ri x** *f* **xa xi** *f* **xh** *f* **xl** *f* **xn** *i* **xt y** *f* **ya yi** *f* **yh** *f* **yl** *f* **yn** *i* **yt**] – Plot a graph
Suppress axes, bold, plotting characters, disconnected, suppress frame, File containing x vector, suppress grid, mark points, region, x origin, suppress x-axis label, x interval, x high bound, x low bound, number of ticks on x-axis, suppress x-axis title, y origin, suppress y-axis label, y interval, y high bound, y low bound, number of ticks on y-axis, suppress y-axis title

title [**-b c l** *string* **v** *string* **u** *string*] – Title a vector or a GPS
title bold, retain case, lower title, upper title, vector title

Generators:

gas [**-c** *i* **if n** *i* **s** *f* **t** *f*] – Generate additive sequence
interval, number, start, terminate

prime [**-c** *i* **h** *i* **l** *i* **n** *i*] – Generate prime numbers
high, low, number

rand [**-c** *i* **h** *f* **l** *f* **m** *f* **n** *i* **s** *i*] – Generate random sequence
high, low, multiplier, number, seed

WARNINGS

Some nodes have a limit on the size of the input vector.

SEE ALSO

graphics(1G), gps(4)

NAME

strace – print STREAMS trace messages

SYNOPSIS

strace [mid sid level] ...

DESCRIPTION

strace without arguments writes all STREAMS event trace messages from all drivers and modules to its standard output. These messages are obtained from the STREAMS log driver [**log**(7)]. If arguments are provided, they must be in triplets of the form *mid, sid, level*, where *mid* is a STREAMS module ID number, *sid* is a sub-id number, and *level* is a tracing priority level. Each triplet indicates that tracing messages are to be received from the given module/driver, sub-id (usually indicating minor device), and priority level equal to or less than the given level. The token *all* may be used for any member to indicate no restriction for that attribute.

The format of each trace message output is as follows:

<seq> <time> <ticks> <level> <flags> <mid> <sid> <text>

<seq>	Trace sequence number
<time>	Time of message in hh:mm:ss
<ticks>	Time of message in machine ticks since boot
<level>	Tracing priority level
<flags>	E : message is also in the error log F : indicates a fatal error N : mail was sent to the system administrator
<mid>	Module ID number of source
<sid>	Sub-id number of source
<text>	Formatted text of the trace message

Once initiated, **strace** will continue to execute until terminated by the user.

EXAMPLES

Output all trace messages from the module or driver whose module ID is 41:

strace 41 all all

Output those trace messages from driver/module ID 41 with sub-ids 0, 1, or 2:

strace 41 0 1 41 1 1 41 2 0

Messages from sub-ids 0 and 1 must have a tracing level less than or equal to 1. Those from sub-id 2 must have a tracing level of 0.

SEE ALSO

log(7)

WARNINGS

Due to performance considerations, only one **strace** process is permitted to open the STREAMS log driver at a time. The log driver has a list of the triplets specified in the command invocation, and compares each potential trace message against this list to decide if it should be formatted and sent up to the **strace** process. Hence, long lists of triplets will have a greater impact on overall STREAMS performance. Running **strace** will have the most impact on

the timing of the modules and drivers generating the trace messages that are sent to the **strace** process. If trace messages are generated faster than the **strace** process can handle them, then some of the messages will be lost. This last case can be determined by examining the sequence numbers on the trace messages output.

NAME

strclean – STREAMS error logger cleanup program

SYNOPSIS

strclean [**–d** logdir] [**–a** age]

DESCRIPTION

strclean is used to clean up the STREAMS error logger directory on a regular basis (for example, by using **cron**(1M)). By default, all files with names matching *error.** in */usr/adm/streams* that have not been modified in the last 3 days are removed. A directory other than */usr/adm/streams* can be specified using the **–d** option. The maximum age in days for a log file can be changed using the **–a** option.

EXAMPLE

The following command has the same result as running **strclean** with no arguments.

strclean –d /usr/adm/streams –a 3

FILES

/usr/adm/streams/error.*

SEE ALSO

cron(1M), strerr(1M)

NOTES

strclean is typically run from **cron**(1M) on a daily or weekly basis.

NAME

strerr – STREAMS error logger daemon

SYNOPSIS

strerr

DESCRIPTION

strerr receives error log messages from the STREAMS log driver [**log**(7)] and appends them to a log file. The error log files produced reside in the directory */usr/adm/streams* and are named *error.mm-dd*, where *mm* is the month and *dd* is the day of the messages contained in each log file.

The format of an error log message is as follows:

<seq> <time> <level> <flags> <mid> <sid> <text>

<seq>	Error sequence number
<time>	Time of message in hh:mm:ss
<level>	Priority level of message for tracing
<flags>	T : the message was also sent to a tracing process F : indicates a fatal error N : send mail to the system administrator
<mid>	Module ID number of source
<sid>	Sub-id number of source
<text>	Formatted text of the error message

Messages that appear in the error log are intended to report exceptional conditions that require the attention of the system administrator. Those messages that indicate the total failure of a STREAMS driver or module should have the F flag set. Those messages requiring the immediate attention of the administrator will have the N flag set, which causes the error logger to send the message to the system administrator via **mail**(1). The priority level usually has no meaning in the error log but will have meaning if the message is also sent to a tracer process.

Once initiated, **strerr** will continue to execute until terminated by the user. Commonly, **strerr** would be executed asynchronously.

WARNINGS

Only one **strerr** process at a time is permitted to open the STREAMS log driver.

If a module or driver is generating a large number of error messages, running the error logger will cause a degradation in STREAMS performance. If a large burst of messages are generated in a short time, the log driver may not be able to deliver some of the messages. This situation is indicated by gaps in the sequence numbering of the messages in the log files.

FILES

/usr/adm/streams/error.*mm-dd*

SEE ALSO

log(7)

NAME

stty – set the options for a terminal

SYNOPSIS

stty [**–a**] [**–g**] [options]

DESCRIPTION

stty sets certain terminal I/O options for the device that is the current standard input. Without arguments, it reports the settings of certain options.

In this report, if a character is preceded by a caret (^), the value of that option is the corresponding Ctrl character (for example, **^h** is Ctrl-H; in this case, recall that Ctrl-H is the same as the Backspace key.) The sequence ^^ means that an option has a null value.

–a Reports all of the option settings.

–g Reports current settings in a form that can be used as an argument to another **stty** command.

Options in the last group are implemented using options in the previous groups. Note that many combinations of options make no sense, but no sanity checking is performed. The options are selected from the following:

Control Modes:

parenb (–parenb) Enable (disable) parity generation and detection.

parodd (–parodd) Select odd (even) parity.

cs5 cs6 cs7 cs8 Select character size (see **termio**(7)).

0 Hang up phone line immediately.

110 300 600 1200 1800 2400 4800 9600 19200 38400
Set terminal baud rate to the number given, if possible. (All speeds are not supported by all hardware interfaces.)

hupcl (–hupcl) Hang up (do not hang up) Dataphone Connection on last close.

hup (–hup) Same as **hupcl (–hupcl)**.

cstopb (–cstopb) Use two (one) stop bits per character.

cread (–cread) Enable (disable) the receiver.

clocal (–clocal) Assume a line without (with) modem control.

loblk (–loblk) Block (do not block) output from a non-current layer.

Input Modes:

ignbrk (–ignbrk) Ignore (do not ignore) break on input.

brkint (–brkint) Signal (do not signal) INTR on break.

ignpar (–ignpar) Ignore (do not ignore) parity errors.

parmrk (–parmrk) Mark (do not mark) parity errors (see **termio**(7)).

inpck (–inpck) Enable (disable) input parity checking.

istrip (–istrip) Strip (do not strip) input characters to seven bits.

inlcr (–inlcr) Map (do not map) NL to CR on input.

igncr (–igncr) Ignore (do not ignore) CR on input.

icrnl (–icrnl) Map (do not map) CR to NL on input.

iuclc (–iuclc) Map (do not map) uppercase alphabetics to lowercase on input.

ixon (–ixon) Enable (disable) START/STOP output control. Output is stopped by sending an ASCII DC3 and started by sending an ASCII DC1.

ixany (–ixany) Allow any character (only DC1) to restart output.

ixoff (–ixoff) Request that the system send (not send) START/STOP characters when the input queue is nearly empty/full.

Output Modes:

opost (–opost) Post-process output (do not post-process output; ignore all other output modes).

olcuc (–olcuc) Map (do not map) lowercase alphabetics to uppercase on output.

onlcr (–onlcr) Map (do not map) NL to CR-NL on output.

ocrnl (–ocrnl) Map (do not map) CR to NL on output.

onocr (–onocr) Do not (do) output CRs at column zero.

onlret (–onlret) On the terminal NL performs (does not perform) the CR function.

ofill (–ofill) Use fill characters (use timing) for delays.

ofdel (–ofdel) Fill characters are DELs (NULs).

cr0 cr1 cr2 cr3 Select style of delay for carriage returns (see **termio**(7)).

nl0 nl1 Select style of delay for line-feeds (see **termio**(7)).

tab0 tab1 tab2 tab3 Select style of delay for horizontal tabs (see **termio**(7)).

bs0 bs1 Select style of delay for backspaces (see **termio**(7)).

ff0 ff1 Select style of delay for form-feeds (see **termio**(7)).

vt0 vt1 Select style of delay for vertical tabs (see **termio**(7)).

Local Modes:

isig (–isig) Enable (disable) the checking of characters against the special control characters INTR, QUIT, and SWTCH.

icanon (–icanon) Enable (disable) canonical input (ERASE and KILL processing).

xcase (–xcase) Canonical (unprocessed) upper/lowercase presentation.

echo (–echo) Echo back (do not echo back) every character typed.

echoe (–echoe) Echo (do not echo) ERASE character as a backspace-space-backspace string. Note: this mode will erase the ERASEed character on many CRT terminals. However, it does not keep track of column position and, as a result, may be confusing on escaped characters, tabs, and

backspaces.

echok (–echok) Echo (do not echo) NL after KILL character.

lfkc (–lfkc) The same as **echok (–echok)**; obsolete.

echonl (–echonl) Echo (do not echo) NL.

noflsh (–noflsh) Disable (enable) flush after INTR, QUIT, or SWTCH.

stwrap (–stwrap) Disable (enable) truncation of lines longer than 79 characters on a synchronous line.

stflush (–stflush) Enable (disable) flush on a synchronous line after every **write**(2).

stappl (–stappl) Use application mode (use line mode) on a synchronous line.

tostop Sends signal SIGTTOU to the process group of a process that tries to write to its controlling terminal if it is not in the foreground process group for that terminal.

iexten Enables **eol2, xcase, ixany, iuclc**, and **olcuc.**

Control Assignments:

control-character c Set *control-character* to *c*, where *control-character* is **erase, kill, intr, quit, swtch, eof, eol, ctab, min, susp,** or **time** (**ctab** is used with **–stappl**; **min** and **time** are used with **–icanon**; see **termio**(7)). If *c* is preceded by an (escaped from the shell) caret (^), then the value used is the corresponding Ctrl character (e.g., **^d** is a Ctrl-D); **^?** is interpreted as DEL and **^–** is interpreted as undefined.

line *i* Set line discipline to *i* ($0 < i < 127$).

start **ixoff** and **ixon** start character.

stop **ixoff** and **ixon** stop character.

Screen Size Assignments:

screen-dimension i Set *screen-dimension* to *i*, where *screen-dimension* is **rows, cols, xpixels,** or **ypixels.**

rows Number of lines on screen.

cols Number of characters per line.

xpixels Dots of resolution per horizontal line.

ypixels Dots of resolution per vertical line.

Combination Modes:

evenp or **parity** Enable **parenb** and **cs7.**

oddp Enable **parenb, cs7,** and **parodd.**

–parity, –evenp, or **–oddp**
Disable **parenb,** and set **cs8.**

raw (–raw or **cooked)**
Enable (disable) raw input and output (no Erase, Kill, INTR, QUIT, SWTCH, EOT, or output post processing).

nl (**–nl**)	Unset (set) **icrnl, onlcr.** In addition **–nl** unsets **inlcr, igncr, ocrnl,** and **onlret.**
lcase (**–lcase**)	Set (unset) **xcase, iuclc,** and **olcuc.**
LCASE (**–LCASE**)	Same as **lcase** (**–lcase**).
tabs (**–tabs** or **tab3**)	Preserve (expand to spaces) tabs when printing.
ek	Reset Erase and Kill characters back to normal # and @.
sane	Resets all modes to some reasonable values.
term	Set all modes suitable for the terminal type *term*, where *term* is one of **tty33, tty37, vt05, tn300, ti700,** or **tek.**

SEE ALSO

tabs(1), ioctl(2), termio(7)

NAME

su – become superuser or another user

SYNOPSIS

su [–] [name [arg ...]]

DESCRIPTION

su allows one to become another user without logging off. The default user *name* is **root** (i.e., superuser).

To use **su**, the appropriate password must be supplied (unless one is already **root**). If the password is correct, **su** will execute a new shell with the real and effective user ID set to that of the specified user. The new shell will be the optional program named in the shell field of the specified user's password file entry (see **passwd**(4)), or **/bin/sh** if none is specified (see **sh**(1)). To restore normal user ID privileges, type an EOF (Ctrl-D) to the new shell.

Any additional arguments given on the command line are passed to the program invoked as the shell. When using programs like **sh**(1), an *arg* of the form **–c** *string* executes *string* via the shell and an arg of **–r** will give the user a restricted shell.

The following statements are true only if the optional program named in the shell field of the specified user's password file entry is like **sh**(1). If the first argument to **su** is a –, the environment will be changed to what would be expected if the user actually logged in as the specified user. This is done by invoking the program used as the shell with an *arg0* value whose first character is –, thus causing first the system's profile (*/etc/profile*) and then the specified user's profile (*.profile* in the new HOME directory) to be executed. Otherwise, the environment is passed along with the possible exception of **$PATH**, which is set to */bin:/etc:/usr/bin* for **root**. Note that if the optional program used as the shell is **/bin/sh**, the user's *.profile* can check *arg0* for **–sh** or **–su** to determine if it was invoked by **login**(1) or **su**(1), respectively. If the user's program is other than **/bin/sh**, then *.profile* is invoked with an *arg0* of *–program* by both **login**(1) and **su**(1).

All attempts to become another user using **su** are logged in the log file */usr/adm/sulog*.

EXAMPLES

To become user **bin** while retaining your previously exported environment, execute this command:

su bin

To become user **bin**, but change the environment to what would be expected if **bin** had originally logged in, execute this command:

su – bin

To execute *command* with the temporary environment and permissions of user **bin**, type this command:

su – bin –c *"command args"*

FILES

/etc/passwd	System password file
/etc/profile	System profile
$HOME/.profile	User profile

/usr/adm/sulog Log file

SEE ALSO

env(1), login(1), sh(1), passwd(4), profile(4), environ(5)

NAME

sum – print checksum and block count of a file

SYNOPSIS

sum [**–r**] file

DESCRIPTION

sum calculates and prints a 16-bit checksum for the named file, and also prints the number of blocks in the file. It is typically used to look for bad spots or to validate a file communicated over a transmission line. The option **–r** causes an alternate algorithm to be used in computing the checksum. The block count will be reported in 512 byte blocks.

SEE ALSO

wc(1)

DIAGNOSTICS

"Read error" is indistinguishable from end of file on most devices; check the block count.

NAME

swap – swap administrative interface

SYNOPSIS

/etc/swap –a swapdev swaplen
/etc/swap –l

DESCRIPTION

swap provides a method of adding, deleting, and monitoring the system swap areas used by the memory manager. The following options are recognized:

–a Add the specified swap area. *swapdev* is the name of the block special device, (e.g. */dev/dsk/zd1s1*). *swaplen* is the length of the swap area in 512-byte blocks. If the *swaplen* is specified as 0, the entire device will be used. If *swaplen* is non-zero, the minimum of the specified value and the actual size of the partition will be used. The starting block number is always 0 (zero). This option can be used only by the superuser. Swap areas are normally added by the system start-up routines in */etc/rc2.d* when going into multiuser mode. Once a device has been allocated for swapping, it cannot be removed until reboot.

–l List the status of all the swap areas. The output has four columns:

DEV The *swapdev* special file for the swap area if one can be found in the */dev/dsk* or */dev* directories, and its major/minor device number in decimal.

LOW The *swaplow* value for the area in 512-byte blocks.

LEN The *swaplen* value for the area in 512-byte blocks.

FREE The number of free 512-byte blocks in the area. If the swap area is being deleted, this column will be marked INDEL.

SEE ALSO

rc(4), rc2(1M)

WARNINGS

No check is done to see if a swap area being added overlaps with an existing swap area or filesystem.

NAME

sweepon, sweepoff – enable/disable patrol seeks on DCC disks

SYNOPSIS

/etc/sweepon [–d] {–a | drivename}
/etc/sweepoff [–d] {–a | drivename}

DESCRIPTION

sweepon enables patrol seeks on one or multiple drives, while **sweepoff** disables them. **sweepon** and **sweepoff** are links to the same program. The **sweepon** utility was created because DCC drive manufacturers recommend that disk heads be moved at regular intervals to increase drive longevity. Therefore, **sweepon** should *always* be invoked at system boot time. The *–a* option causes patrol seeks to be enabled/disabled on all configured DCC drives. Alternatively, a *drivename* can be specified which will cause patrol seeks to be enabled or disabled only on the indicated drive. *drivename* may be a simple disk name such as *zd0* or *zd4,* without any path prefix, or it may be a full pathname. If the first form is used, the appropriate path to the special device is generated automatically.

If the **–d** option is specified, debug output is produced on **stdout.**

When the *–a* option is used with **sweepon,** it is possible to cause devices to be skipped (so that patrol seeks are *not* enabled) by placing the simple device name of the drive in the file */etc/sweepon.ignore.* The format of the *sweepon.ignore* file is one disk name per line. The name should start in the first column. Comment lines may be specified with a pound sign (#) in the first column. Blank lines are ignored. For example, if you wish to enable patrol seeks on all DCC drives except **zd2,** you would add *zd2* to the *sweepon.ignore* file on its own line, and that device will be skipped when **sweepon** executes. This file is only looked at when the *–a* option is used, and only for **sweepon.** It is recommended that patrol seeks remain enabled on all DCC drives.

FILES

/etc/sweepon.ignore

DIAGNOSTICS

The diagnostics produced by **sweepon** are intended to be self-explanatory.

NAME

sync – update the superblock

SYNOPSIS

sync

DESCRIPTION

sync executes the **sync** system primitive. If the system is to be stopped, **sync** must be called to insure filesystem integrity. It will flush all previously unwritten system buffers out to disk, thus assuring that all file modifications up to that point will be saved. See **sync**(2) for details.

SEE ALSO

sync(2)

NAME

sysadm – menu interface to do system administration

SYNOPSIS

sysadm [*subcommand*]

DESCRIPTION

This command, when invoked without an argument, presents a menu of system administration subcommands, from which the user selects. If the optional argument is presented, the named subcommand is run or the named submenu is presented.

The **sysadm** command can be given a password. See **admpasswd** in the **Subcommands** section.

Subcommands

The following menus of subcommands are available. (The number of bullets (•) in front of each item indicates the level of the menu or subcommand.)

• diskmgmt
disk management menu

The subcommands on this menu allow you to check, mount, and unmount filesystems.

• • checkfsys
check a filesystem for errors

This procedure checks a filesystem for errors. If there are errors, this procedure attempts to repair them.

• • harddisk
hard disk management menu

The subcommand on this menu allows you to view the partitions on a disk.

• • • display
display disk partitioning

This procedure displays current disk partitioning information.

• • mountfsys
mount a filesystem

This procedure mounts a filesystem, making it available to users. The filesystem must be unmounted with the **umountfsys** command.

• • umountfsys
unmount a filesystem

This procedure unmounts a filesystem. **umountfsys** can be used only to unmount filesystems mounted with the **mountfsys** command.

• filemgmt
file management menu

The subcommands on this menu allow you to back up and restore files. Subcommands are also provided to determine which files might be best kept on disk based on age or size.

• • backup
backup files from disk to tape

This procedure saves copies of files from filesystems to tape. There are two kinds of backups:

COMPLETE – copies all files (useful in case of serious filesystem damage)

INCREMENTAL – copies files changed since the last backup

The normal usage is to do a complete backup of each filesystem and then periodically do incremental backups. Two cycles are recommended (one set of complete backups and several incrementals to each cycle). Files backed up with **backup** are restored using **restore**.

• • bupsched
backup reminder scheduling menu

Backup scheduling is used to schedule backup reminder messages and backup reminder checks. Backup reminder messages are sent to the console to remind the administrator to back up particular filesystems when the machine is shut down or a reminder check has been run during the specified time period.

Backup reminder checks specify particular times at which the system will check to see if any backup reminder messages have been scheduled.

• • • schedcheck
schedule backup reminder checks

Backup reminder checks are run at specific times to see if any reminders are scheduled. The user specifies the times at which the check is to be run. Checks are run for the reminder messages scheduled by **schedmsg**.

• • • schedmsg
schedule backup reminder message

Backup reminder messages are sent to the console if the machine is shut down or a reminder check has been scheduled. The user specifies the both times at which it is appropriate to send a message and the filesystems to be included in the message.

• • diskuse
display how much disk space is being used

This procedure displays the percentage of disk space currently occupied by files. The list is organized by filesystem names.

- • fileage
  list files older than a particular date

  fileage prints the names of all files older than the specified date. If no date is entered, all files older than 90 days will be listed. If no directory is specified, the */usr/admin* directory will be used.

- • filesize
  list the largest files in a particular directory

  filesize prints the names of the largest files in a specific directory. If no directory is specified, the */usr/admin* directory will be used. If you don't specify the number of files to print, 10 files will be listed.

- • restore
  restore files from **backup** and **store** media to disk

  restore copies files from tapes made by **backup** and **store** back onto disk. You can restore individual files, directories of files, or the entire contents of a tape. You can restore from both "incremental" and "complete" media. You can also list the names of files stored on the tape.

- • store
  store files and directories of files onto tape

  store copies files from disk to tape and allows you to optionally verify that they worked and to optionally remove them when done. Typically, these would be files that you want to archive or restrict access to. You can store single files and directories of files. Use the **restore** command to place stored files back on disk and to list the files stored.

- machinemgmt
  machine management menu

  Machine management functions are tools used to operate the machine (such as rebooting or going to the firmware monitor).

- • firmware
  stop all running programs then enter firmware mode

  This procedure stops all running programs, closes any open files, writes out information to disk (such as directory information), then enters the firmware mode. (Machine diagnostics and other special functions that are not available on the operating system can be run in firmware mode.)

- • reboot
  stop all running programs then reboot the machine

  reboot stops all running programs, closes any open files, writes out information to disk (such as directory information), and then reboots the machine. This procedure can be used to get out of some types of system trouble, such as when a process cannot be killed.

• • whoson
print list of users currently logged onto the system

whoson prints the login ID, terminal device number, and sign-on time of all users who are currently using the computer.

• packagemgmt
package management

These submenus and subcommands manage various software packages such as **lp** and **uucp**.

• softwaremgmt
software management menu

These subcommands permit the user to install, list, or remove software. The "remove" capability is dependent on the software package. See the instructions delivered with each package.

• • installpkg
install new software package onto disk

install copies files onto disk and performs additional work if necessary so that the software can be run.

• • listpkg
list packages already installed

This subcommand shows you a list of currently installed software packages.

• • removepkg
remove previously installed package from disk

This subcommand displays a list of currently installed software packages. The actions necessary to remove the software packages you specify will then be performed.

• syssetup
system setup menu

System setup routines allow the user to tell the computer what its environment looks like: what the date, time, and time zone are, what administration and system capabilities are to be under password control, what the machine's name is, and so on. The first-time setup sequence is also here.

• • admpasswd
assign or change administrative passwords

admpasswd lets you set or make changes to passwords for administrative commands and logins such as **setup** and **sysadm**.

• • datetime
set the date, time, time zone, and daylight savings time

datetime tells the computer the date, time, time zone, and whether you observe Daylight Savings Time (DST). It is normally run once when the machine is first set up. If you observe DST, the computer will automatically start to observe it in the spring and return to Standard Time in the fall. The machine has to be turned off and turned back on again to guarantee that ALL times will be reported correctly. Most are correct the next time the user logs in.

- • nodename
  set the node name of this machine

  This allows you to change the node name of this machine. The node name is used by various communications networks to identify this machine.

- • setup
  set up your machine the very first time

  setup allows you to define the first login, to set the passwords on the user-definable administration logins, and to set the time zone for your location.

- • syspasswd
  assign system passwords

  syspasswd lets you set system passwords normally reserved for the very knowledgeable user. For this reason, this procedure can assign those passwords, but cannot change or clear them. Once set, they can only be changed by the specific login or the **root** login.

- ttymgmt
  terminal management

  This procedure allows you to manage the computer's terminal functions.

- • lineset
  show tty line settings and hunt sequences

  The tty line settings are often hunt sequences where, if the first line setting does not work, the line "hunts" to the next line setting until it reaches one that does work. This subcommand shows the various sequences with only the names of the line settings they contain. It also shows each line setting in detail.

- • mklineset
  create new tyy line settings and hunt sequences

  This subcommand helps you to create entries for tty line settings. You might want to add line settings that are not in the current set or to create hunt sequences with only specific line settings in them. The created hunt sequences are circular; stepping past the last setting puts you on the first.

• • modtty
show and optionally modify characteristics of tty lines

This subcommand reports and allows you to change the characteristics of tty lines (also called "ports").

• usermgmt
user management menu

These subcommands allow you to add, modify and delete the list of users that have access to your machine. You can also place them in separate groups so that they can share access to files within the group but protect themselves from other groups.

• • addgroup
add a group to the system

addgroup adds a new group name or ID to the computer. Group names and IDs are used to identify groups of users who desire common access to a set of files and directories.

• • adduser
add a user to the system

adduser installs a new login ID on the machine. You are asked a series of questions about the user and then the new entry is made. You can enter more than one user at a time. Once this procedure is finished, the new login ID is available.

• • delgroup
delete a group from the system

delgroup allows you to remove groups from the computer. The deleted group is no longer identified by name. However, files can still be identified with the group ID number.

• • deluser
delete a user from the system

deluser allows you to remove users from the computer. The deleted user's files are removed from disk and their logins are removed from the */etc/passwd* file.

• • lsgroup
list groups in the system

lsgroup lists all groups that have been entered into the computer. This list is updated automatically by **addgroup** and **delgroup**.

• • lsuser
list users in the system

lsuser lists all users that have been entered into the computer. This list is updated automatically by **adduser** and **deluser**.

• • modadduser
modify defaults used by adduser

modadduser allows you to change some of the defaults used when **adduser** creates a new login. Changing the defaults does not affect any existing logins, only logins made from this point on.

• • modgroup
make changes to a group on the system

modgroup allows you to change the name of a group.

• • moduser
menu of commands to modify a user's login

This menu contains commands that modify the various aspects of a user's login.

• • • chgloginid
change a user's login ID

This procedure allows you to change a user's login ID. Administrative and system logins cannot be changed.

• • • chgpasswd
change a user's passwd

This procedure allows removal or change of a user's password. Administrative and system login passwords cannot be changed. To change administrative and system login passwords, see the system setup menu: **sysadm syssetup**.

• • • chgshell
change a user's login shell

This procedure allows you to change the command run when a user logs in. The login shell of the administrative and system logins cannot be changed by this procedure.

EXAMPLES

sysadm adduser

FILES

The files that support **sysadm** are found in */usr/admin*.

The menu starts in the directory */usr/admin/menu*.

NAME

tabs – set tabs on a terminal

SYNOPSIS

tabs [tabspec] [**–T**type] [**+mn**]

DESCRIPTION

tabs sets the tab stops on the user's terminal according to the tab specification *tabspec*, after clearing any previous settings. The user's terminal must have hardware tabs that can be set remotely.

tabspec Four types of tab specification are accepted for *tabspec*:

canned	(*–code*)
repetitive	(*–n*)
arbitrary	(*n1,n2,...*)
file	(*––file*)

If no *tabspec* is given, the default value is **–8** (i.e., "standard" tabs). The lowest column number is 1. Note that for **tabs**, column 1 always refers to the leftmost column on a terminal, even one whose column markers begin at 0, such as the DASI 300, DASI 300s, and DASI 450.

–code Use one of the codes listed below to select a *canned* set of tabs. The legal codes and their meanings are as follows:

–a 1,10,16,36,72
Assembler, IBM S/370, first format

–a2 1,10,16,40,72
Assembler, IBM S/370, second format

–c 1,8,12,16,20,55
COBOL, normal format

–c2 1,6,10,14,49
COBOL compact format (columns 1-6 omitted). Using this code, the first typed character corresponds to card column 7, one space gets you to column 8, and a tab reaches column 12. Files using this tab setup should include a format specification as follows (see **fspec**(4)):

```
<:t-c2 m6 s66 d:>
```

–c3 1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67
COBOL compact format (columns 1-6 omitted), with more tabs than **–c2**. This is the recommended format for COBOL. The appropriate format specification is as follows (see **fspec**(4)):

```
<:t-c3 m6 s66 d:>
```

–f 1,7,11,15,19,23
FORTRAN

–p 1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61
PL/I

–s 1,10,55
SNOBOL

–u 1,12,20,44
UNIVAC 1100 Assembler

–n A *repetitive* specification requests tabs at columns 1+*n*, 1+2*n*, etc. Of particular importance is the value 8; this represents the "standard" tab setting and is the most likely tab setting to be found at a terminal. Another special case is the value **0**, implying no tabs at all.

n1,n2,... The *arbitrary* format permits the user to type any chosen set of numbers, separated by commas, in ascending order. Up to 40 numbers are allowed. If any number (except the first one) is preceded by a plus sign, it is taken as an increment to be added to the previous value. Thus, the formats **1,10,20,30**, and **1,10,+10,+10** are considered identical.

––file If the name of a *file* is given, **tabs** reads the first line of the file, searching for a format specification (see **fspec**(4)). If it finds one there, it sets the tab stops according to it, otherwise it sets them as **–8**. This type of specification may be used to make sure that a tabbed file is printed with correct tab settings and would be used with the **pr**(1) command:

tabs *––file*; **pr** *file*

Any of the following also may be used. If a given flag occurs more than once, the last value given takes effect:

–T*type* **tabs** usually needs to know the type of terminal in order to set tabs and always needs to know the type to set margins. *type* is a name listed in **term**(5). If no **–T** flag is supplied, **tabs** uses the value of the environment variable **TERM**. If **TERM** is not defined in the environment (see **environ**(5)), **tabs** tries a sequence that will work for many terminals.

+m*n* The margin argument may be used for some terminals. It causes all tabs to be moved over *n* columns by making column *n+1* the left margin. If **+m** is given without a value of *n*, the value assumed is **10**. For a TermiNet, the first value in the tab list should be **1**, or the margin will move even further to the right. The normal (leftmost) margin on most terminals is obtained by **+m0**. The margin for most terminals is reset only when the **+m** flag is given explicitly.

Tab and margin setting is performed via the standard output.

EXAMPLES

tabs –a Example of using *–code* (*canned* specification) to set tabs to the settings required by the IBM assembler: columns 1, 10, 16, 36, 72.

tabs –8 Example of using *–n* (*repetitive* specification), where *n* is 8, causes tabs to be set every eighth position: 1+(1*8), 1+(2*8), ... which evaluate to columns 9, 17, ...

tabs 1,8,36 Example of using *n1,n2,...* (*arbitrary* specification) to set tabs at columns 1, 8, and 36.

tabs --$HOME/fspec.list/att4425
Example of using *--file* (*file* specification) to indicate that tabs should be set according to the first line of *$HOME/fspec.list/att4425* (see **fspec**(4)).

DIAGNOSTICS

illegal tabs	Arbitrary tabs are ordered incorrectly.
illegal increment	A zero or missing increment is found in an arbitrary specification.
unknown tab code	A *canned* code cannot be found.
can't open	*--file* option was used and file can't be opened.
file indirection	*--file* option was used and the specification in that file points to yet another file. Indirection of this form is not permitted.

SEE ALSO
newform(1), pr(1), tput(1), fspec(4), terminfo(4), environ(5), term(5)

NOTE
There is no consistency among different terminals regarding ways of clearing tabs and setting the left margin.

tabs clears only 20 tabs (on terminals requiring a long sequence), but is willing to set 64.

WARNINGS
The *tabspec* used with the **tabs** command is different from the one used with the **newform**(1) command. For example, **tabs –8** sets every eighth position; whereas **newform –i–8** indicates that tabs are set every eighth position.

NAME

tail – deliver the last part of a file

SYNOPSIS

tail [±[number][**lbc**[**f**]]] [file]

DESCRIPTION

tail copies the named file to the standard output beginning at a designated place. If no file is named, the standard input is used.

Copying begins at distance *+number* from the beginning of the input or *–number* from the end (if *number* is null, the value 10 is assumed). *Number* is counted in units of lines, blocks, or characters, according to the appended option **l**, **b**, or **c**. When no units are specified, counting is by lines.

With the **–f** ("follow") option, if the input file is not a pipe, the program will not terminate after the line of the input file has been copied, but will enter an endless loop, wherein it sleeps for a second and then attempts to read and copy further records from the input file. Thus, it may be used to monitor the growth of a file that is being written by some other process. For example, the following command prints the last 10 lines of the file *fred*, followed by any lines that are appended to *fred* between the time **tail** is initiated and killed.

tail –f fred

As another example, the next command prints the last 15 characters of the file *fred*, followed by any lines that are appended to *fred* between the time **tail** is initiated and killed.

tail –15cf fred

SEE ALSO

dd(1M)

WARNINGS

The **tail** command will tail only the last 4096 bytes of a file regardless of its line count.

BUGS

Tails relative to the end of the file are stored in a buffer, and thus are limited in length. Various kinds of anomalous behavior may happen with character special files.

NAME

tar – tape file archiver

SYNOPSIS

/bin/tar –c[vwfbh[#s]] *device block files ...*
/bin/tar –r[vwb[#s]] *device block [files ...]*
/bin/tar –t[vf[#s]] *device*
/bin/tar –u[vwb[#s]] *device block [files ...]*
/bin/tar –x[lmovwf[#s]] *device [files ...]*

DESCRIPTION

tar saves and restores files on magnetic tape. Its actions are controlled by the key argument. The key is a string of characters containing one function letter (**c**, **r**, **t**, **u**, or **x**) and possibly followed by one or more function modifiers (**v**, **w**, **f**, **b**, and **#**). Other arguments to the command are *files* (or directory names) specifying which files are to be dumped or restored. In all cases, appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

The tape is written using the "extended tar format" defined by the POSIX 1003.1 standard. Tapes can consist of multiple volumes; for more information, see the discussion following the list of command arguments. This **tar** reads tapes written by BSD and ATT **tar** programs as well as other tapes conforming to the POSIX format. Tapes written in the extended tar format can be read by BSD **tar** as long as only directories and regular files are placed on the tape. For compatibility with older tar progams not adhering to the POSIX specification, the program **otar** is supplied. This program writes tapes in the format expected by ATT **tar** programs.

POSIX compliance requires that the user's **umask** value shall modify file creation modes unless the root user is executing the program. The AT&T semantics were such that **tar** would set the **umask** value to 0 before creating files. This is now done only when the root user executes the program. Non-root users wishing to have files extracted with the same modes as appear on tape must set the **umask** value to 0 before executing the **tar** program.

The function portion of the key is specified by one of the following letters:

r Replace. The named *files* are written on the end of the tape. The **c** function implies this function.

x Extract. The named *files* are extracted from the tape. If a named file matches a directory whose contents had been written onto the tape, this directory is (recursively) extracted. Use the file or directory's relative path when appropriate, or **tar** will not find a match. The owner, modification time, and mode are restored (if possible). If no *files* argument is given, the entire content of the tape is extracted. Note that if several files with the same name are on the tape, the last one overwrites all earlier ones.

t Table. The names and other information for the specified files are listed each time that they occur on the tape. The listing is similar to the format produced by the **ls –l** command. If no *files* argument is given, all names on the tape are listed.

u Update. The named *files* are added to the tape if they are not already there or have been modified since last written on that tape. This key implies the **r** key.

c Create a new tape; writing begins at the beginning of the tape, instead of after the last file. This key implies the **r** key.

The characters below may be used in addition to the letter that selects the desired function. Use them in the order shown in the synopsis.

#s This modifier determines the drive on which the tape is mounted (replace # with the drive number) and the speed of the drive (replace **s** with **l**, **m**, or **h** for low, medium or high). The modifier tells **tar** to use a drive other than the default drive (or the drive specified with the **–f** option). For example, with the **5h** modifier, **tar** would use */dev/rmt/xt0h* or */dev/rmt/ts0h* instead of the default drives */dev/rmt/xt0* or */dev/rmt/ts0*, respectively. However, if **–f /dev/rmt/tm0** appeared on the command line, **tar** would use */dev/rmt/tm0*.

v Verbose. Normally, **tar** does its work silently. The **v** (verbose) option causes it to type the name of each file it treats, preceded by the function letter. With the **t** function, **v** gives more information about the tape entries than just the name.

w What. This causes **tar** to print the action to be taken, followed by the name of the file, and then wait for the user's confirmation. If a word beginning with **y** is given, the action is performed. Any other input means "no". This is not valid with the **t** key.

f File. This causes **tar** to use the *device* argument as the name of the archive instead of */dev/rmt/xt0* or */dev/rmt/ts0*. If the name of the file is **–**, **tar** writes to the standard output or reads from the standard input, whichever is appropriate. Thus, **tar** can be used as the head or tail of a pipeline. **tar** can also be used to move hierarchies with the command:

cd *fromdir*; **tar cf – . | (cd** *todir*; **tar xf –)**

b Blocking Factor. This causes **tar** to use the *block* argument as the blocking factor for tape records. The default is 1, the maximum is 20. This function should not be supplied when operating on regular archives or block special devices. It is mandatory however, when reading archives on raw magnetic tape archives (see **f** above). The block size is determined automatically when reading tapes created on block special devices (key letters **x** and **t**).

l Link. This tells **tar** to complain if it cannot resolve all of the links to the files being dumped. If **l** is not specified, no error messages print.

m Modify. This tells **tar** not to restore the modification times. The modification time of the file will be the time of extraction.

o Ownership. This causes extracted files to take on the user and group identifier of the user running the program, rather than those on tape. This is only valid with the **x** key.

h Force **tar** to follow symbolic links as if they were normal files or directories. Normally, **tar** does not follow symbolic links.

In compliance with the POSIX specification, the **tar** program provides support for multiple volumes. If the end of tape is reached, **tar** prompts for a new device. Mount the next tape, then press Return at the prompt. The program will then open the device last used for I/O and continue reading or writing.

If you want to abort the read or write, enter the letter **q** to terminate the **tar** program. If you were writing to the tape, the data up to the last file placed on the tape will be readable. The file being written when the end of tape was reached will be incomplete.

When using multi-volume tapes with this program, it is guaranteed that the final block on the tape will contain an even multiple of 512 bytes. However, the block might be smaller than the block size used on the rest of the tape. This program will accept other POSIX-compliant 1/2-inch **tar** tapes as long as the final block on the tape is written in an integer multiple of 512 bytes.

Cartridge tapes should not be considered interchangable with non-Sequent machines. For interchange with non-Sequent machines use 1/2 inch tapes. With the exception of the final file on the tape, a **tar** program that does not support multiple volumes should be able to read the first reel of a multi-volume tar tape. The remainder of the tapes are not usable. For this reason, if you are creating tapes to be used on a machine that might not be POSIX-compliant, you should place all information on a single tape.

FILES

/dev/rmt/*

SEE ALSO

ar(1), cpio(1), ls(1)

DIAGNOSTICS

Complains about bad key characters and tape read/write errors.

Complains if not enough memory is available to hold the link tables.

BUGS

There is no way to ask for the *n*-th occurrence of a file.

Tape errors are handled ungracefully.

The **u** option can be slow.

The **b** option should not be used with archives that are going to be updated. The current magnetic tape driver cannot backspace raw magnetic tape. If the archive is on a disk file, the **b** option should not be used at all, because updating an archive stored on disk can destroy it.

otar limits filenames to 100 characters. The POSIX format used by **tar** extends this limit to 255 characters. However, to achieve this limit, a "/" in the pathname would have to occur at 156 characters from the start of the filename. Therefore, it is likely that less than the theoretical maximum filename length can be preserved.

otar will not copy empty directories and special files. **tar** does write both of these to tape.

NAME

tasktbl – create security table in shared memory

SYNOPSIS

tasktbl

DESCRIPTION

The Create Security Table (**tasktbl**) command is used to create an in-memory version of the security information in the global security table *($TFADMFILE/etc/task)*. This command speeds access to critical information and allows the menu program to quickly and efficiently resolve security references as each menu is displayed.

The ptx/ADMIN menu system uses the security table to verify if a user can access a menu option. **tasktbl** is usually called by the menu system (**menu**) when the system fails to access the security table. When **tasktbl** is called, the system displays a message on the screen to advise the user that it is regenerating the security table.

Several commands in the menu system (for example, Add a User Account) remove the in-memory copy of the security table to prevent use of invalid or out-of-date security information. The next user to display a menu and select a command will automatically call the **tasktbl** program and trigger the regeneration of the security table.

The system password and group files (*/etc/passwd* and */etc/group*) as well as the disk based version of the security table *($TFADMFILE/etc/task)* store the information used to create the in-memory version of the security table. If any of these three files are altered, use the Delete Security Table command (**dltsectbl**) to remove the in-memory copy of the table. The menu system will generate a new copy when it is required.

If security information is static, you can add a call to the **tasktbl** program to the system startup routine. This call will generate the security table as part of the general system initialization.

FILES

${TFADMFILE}/etc/task	Global task table
/etc/group	System group file
/etc/passwd	System password file

SEE ALSO

dltsectbl(1M), menu(1M), tfipc(1M)

NOTES

This command is usually called by the menu system.

NAME

tee – pipe fitting

SYNOPSIS

tee [**–i**] [**–a**] [file] ...

DESCRIPTION

tee transcribes the standard input to the standard output and makes copies in the *files*. The options for this command are as follows:

–i Ignore interrupts.

–a Cause the output to be appended to the *files* rather than overwriting them.

NAME

test – condition evaluation command

SYNOPSIS

Bourne and Korn Shell

test
expr
[expr]

C Shell

/bin/test
expr

DESCRIPTION

Bourne and Korn Shell

test evaluates the expression *expr* and, if its value is true, sets a zero (true) exit status; otherwise, a non-zero (false) exit status is set. **test** also sets a non-zero exit status if there are no arguments. When permissions are tested, the effective user ID of the process is used.

All operators, flags, and brackets (brackets used as shown in the second **SYNOPSIS** line) must be separate arguments to the **test** command. Normally these items are separated by spaces.

The following primitives are used to construct *expr*:

-r *file*	True if *file* exists and is readable.
-w *file*	True if *file* exists and is writable.
-x *file*	True if *file* exists and is executable.
-f *file*	True if *file* exists and is a regular file.
-d *file*	True if *file* exists and is a directory.
-c *file*	True if *file* exists and is a character special file.
-b *file*	True if *file* exists and is a block special file.
-p *file*	True if *file* exists and is a named pipe (fifo).
-u *file*	True if *file* exists and its set-user-ID bit is set.
-g *file*	True if *file* exists and its set-group-ID bit is set.
-k *file*	True if *file* exists and its sticky bit is set.
-s *file*	True if *file* exists and has a size greater than zero.
-t [*fildes*]	True if the open file whose file descriptor number is *fildes* (1 by default) is associated with a terminal device.
-z *s1*	True if the length of string *s1* is zero.
-n *s1*	True if the length of the string *s1* is non-zero.
s1 = *s2*	True if strings *s1* and *s2* are identical.
s1 != *s2*	True if strings *s1* and *s2* are *not* identical.
s1	True if *s1* is *not* the null string.

n1 **–eq** *n2* True if the integers *n1* and *n2* are algebraically equal. Any of the comparisons **–ne**, **–gt**, **–ge**, **–lt**, and **–le** may be used in place of **–eq**.

These primaries can be combined with the following operators:

! Unary negation operator.

–a Binary *and* operator.

–o Binary *or* operator (**–a** has higher precedence than **–o**).

(expr **)** Parentheses for grouping. Parentheses are meaningful to the shell and, therefore, must be quoted.

C Shell

/bin/test evaluates the expression *expr*, and if its value is true then returns zero exit status; otherwise, a non zero exit status is returned. **/bin/test** returns a non zero exit if there are no arguments.

The following primitives are used to construct *expr*.

–r file true if the file exists and is readable.

–w file true if the file exists and is writable.

–f file true if the file exists and is not a directory.

–d file true if the file exists and is a directory.

–s file true if the file exists and has a size greater than zero.

–t [fildes]
true if the open file whose file descriptor number is *fildes* (1 by default) is associated with a terminal device.

–z s1 true if the length of string *s1* is zero.

–n s1 true if the length of the string *s1* is nonzero.

s1 = s2 true if the strings *s1* and *s2* are equal.

s1 != s2 true if the strings *s1* and *s2* are not equal.

s1 true if *s1* is not the null string. This will only work if no other primitives are used in *expr*.

n1 **–eq** n2
true if the integers *n1* and *n2* are algebraically equal. Any of the comparisons **–ne**, **–gt**, **–ge**, **–lt**, or **–le** may be used in place of **–eq**.

These primaries may be combined with the following operators:

! unary negation operator

–a binary *and* operator

–o binary *or* operator

(expr **)**
parentheses for grouping.

–a has higher precedence than **–o**. Notice that all the operators and flags are separate arguments to *test*. Notice also that parentheses are meaningful to the Shell and must be escaped.

The left bracket ([ ) is a shell alias for the *test* command. This lets you create shell scripts using the following construct:

if [<conditional expr>] ; then ...

SEE ALSO

csh(1), find(1), ksh(1), sh(1)

WARNINGS

If you test a file you own (the **-r**, **-w**, or **-x** tests), but the permission tested does not have the *owner* bit set, a non-zero (false) exit status will be returned even though the file may have the *group* or *other* bit set for that permission. The correct exit status will be set if you are superuser.

The = and != operators have a higher precedence than the **-r** through **-n** operators, and = and != always expect arguments. Therefore, = and != cannot be used with the **-r** through **-n** operators.

If more than one argument follows the **-r** through **-n** operators, only the first argument is examined. The others are ignored, unless **-a** or **-o** is the second argument.

NAME

tfipc – interprocess communication reporter

SYNOPSIS

tfipc [**–r**] [**–s**] [**–d**]

DESCRIPTION

The Interprocess Communication Reporter (**tfipc**) command identifies and displays all interprocess communication items used by the ptx/ADMIN menu system. This includes message queues for sending messages, shared memory for the security table, and semaphores that control access to the shared memory segment.

–r This instructs the Interprocess Communication Reporter to identify and **remove** all menu system related message queues, shared memory segments, and semaphores. The remove flag is optional.

–s Silent mode instructs the reporter to omit the display of the ids and explanation of each ipc element. This is generally used in conjunction with the **remove** option to silently remove all ipc components. Selection of silent mode is optional and relevant only when coupled with the remove option.

–d Detail mode instructs the reporter to display the ids, explanation, and detailed information for each ipc element. Selection of detail mode is optional.

SEE ALSO

menu(1M), menumsg(1M), tasktbl(1M)

NOTES

When you remove the message queues, any messages in the queue are also removed.

NAME

tfmenush – assign menu system as user login shell

SYNOPSIS

tfmenush

DESCRIPTION

The **tfmenush** command lets you assign the menu system as a user's login shell. When you use this command, the program **/bin/sh** reads commands from the system profile */etc/profile* and from the *.profile* in the user's home directory. The command then executes the menu system, **menuprom,** located in the *$TFBIN* directory.

EXAMPLES

A System Administrator installs the menu system in the */usr/tf* directory. To ensure that a user enters the menu system when they logon, the System Administrator must enter the full pathname of the **tfmenush** program in the user's entry in the */etc/passwd* file.

The following example is a typical entry in the */etc/passwd* for menu system user Bill:

```
bill::101:100:Billy Boy:/u/bill:/usr/tf/bin/tfmenush
```

To ensure that **tfmenush** knows where to find the menu system (**menu**), the System Administrator enters the following line in the menu system Environment File */etc/tfenv:*

```
TFBIN=/usr/tf/bin
```

If this environment variable is not set, the login process will fail and the user cannot log onto the system.

The following steps occur during the login process:

1. The login process spawns a Bourne shell that runs commands found in */etc/profile* and *$HOME/.profile.*

   The commands in */etc/profile* establish the value of the environment variable **TFBIN.** The environment variable **TERM** is generally set in the *$HOME/.profile* file. To help avoid conflict with other software, the menu system can use the environment variable **TFTERM** to set the terminal type. **TFTERM** is set in the user's *$HOME/.profile.*

2. The login shell turns control over to **tfmenush.**

3. **tfmenush** resets the value of the **SHELL** environment variable to **/bin/sh** and executes the menu system, **$TFBIN/menu.** Note that unless the **TFBIN** environment variable is set in the */etc/profile,* **tfmenush** fails at this point and log the user off the system.

Environment Variables (set in */etc/tfenv*)

TFBIN Contains the name of the directory where the menu system standard binaries are stored.

TFADMBIN Contains the name of the directory where the System Administrator's binaries are stored. In many installations, **TFBIN** and **TFADMBIN** are set to the same value.

TFADMFILE Contains the name of the head directory where menu system subdirectories are stored. There are four important directories:

etc	Control information
misc	Error logs and dead messages
edc	Compiled forms (in most cases)
menu	Menu definitions (in most cases)

FILES

/etc/passwd	System password file
/etc/tfenv	Establishes **TFBIN** environment variable
$HOME/.profile	Establishes **TERM** and/or **TFTERM** variables

SEE ALSO

login(1), menu(1M), passwd(4)

NAME

tfnotify – send completion message

SYNOPSIS

tfnotify [*message*]

DESCRIPTION

A program can use the The Send Completion Message (**tfnotify**) command to send a message to a user. If there is an active menu process at the user's terminal, **tfnotify** places the message in the ptx/ADMIN menu system menu message queue. If there is not an active menu process at the terminal, the message is displayed on the standard error.

This command is primarily used to send completion messages to a user from a shell command or from within a form template.

message Text of the message sent to the message queue or displayed on the standard error. If message text is not entered on the command line, the standard input is read. Message text is required.

Environment Variables

TFUSER Contains the name of the user who will receive the message. Set by the menu process.

TFTTY Contains the name of the terminal that will receive the message. Set by the menu process.

FILES

/etc/passwd	System password file
${TFADMFILE}/menumsg/*	System message files
${TFADMFILE}/misc/dead.msg	System dead message file

SEE ALSO

menu(1M), menumsg(1M)

NAME

tic – terminfo compiler

SYNOPSIS

tic [–v[n]] [–c] file

DESCRIPTION

tic translates a **terminfo**(4) file from the source format into the compiled format. The results are placed in the directory */usr/lib/terminfo*. The compiled format is necessary for use with the library routines described in **curses**(3X).

–v*n* (Verbose) output to standard error trace information showing **tic**'s progress. The optional integer *n* is a number from 1 to 10, inclusive, indicating the desired level of detail of information. If *n* is omitted, the default level is 1. If *n* is specified and greater than 1, the level of detail is increased.

–c Only check *file* for errors. Errors in **use=** links are not detected.

file Contains one or more **terminfo** terminal descriptions in source format (see **terminfo**(4)). Each description in the file describes the capabilities of a particular terminal. When a use=*entry-name* field is discovered in a terminal entry currently being compiled, **tic** reads in the binary from */usr/lib/terminfo* to complete the entry. (Entries created from *file* will be used first. If the environment variable **TERMINFO** is set, that directory is searched instead of */usr/lib/terminfo*.) **tic** duplicates the capabilities in *entry-name* for the current entry, with the exception of those capabilities that explicitly are defined in the current entry.

If the environment variable **TERMINFO** is set, the compiled results are placed there instead of */usr/lib/terminfo*.

FILES

/usr/lib/terminfo/?/* Compiled terminal description database

SEE ALSO

curses(3X), term(4), terminfo(4)

DIAGNOSTICS

Most diagnostic messages produced by **tic** during the compilation of the source file are preceded with the approximate line number and the name of the terminal currently being worked on.

mkdir ... returned bad status
The named directory could not be created.

File does not start with terminal names in column one
The first thing seen in the file, after comments, must be the list of terminal names.

Token after a seek(2) not NAMES
Somehow the file being compiled changed during the compilation.

Not enough memory for use_list element
or
Out of memory
Not enough free memory was available (**malloc**(3) failed).

Can't open ...
The named file could not be created.

Error in writing ...
The named file could not be written to.

Can't link ... to ...
A link failed.

Error in re-reading compiled file ...
The compiled file could not be read back in.

Premature EOF
The current entry ended prematurely.

Backspaced off beginning of line
This error indicates something wrong happened within **tic**.

Unknown Capability - "..."
The named invalid capability was found within the file.

Wrong type used for capability "..."
For example, a string capability was given a numeric value.

Unknown token type
Tokens must be followed by '@' to cancel, ',' for booleans, '#' for numbers, or '=' for strings.

"...": bad term name
or
Line ...: Illegal terminal name - "..."
Terminal names must start with a letter or digit
The given name was invalid. Names must not contain white space or slashes, and must begin with a letter or digit.

"...": terminal name too long.
An extremely long terminal name was found.

"...": terminal name too short.
A one-letter name was found.

"..." filename too long, truncating to "..."
The given name was truncated to 14 characters due to filename length limitations.

"..." defined in more than one entry. Entry being used is "...".
An entry was found more than once.

Terminal name "..." synonym for itself
A name was listed twice in the list of synonyms.

At least one synonym should begin with a letter.
At least one of the names of the terminal should begin with a letter.

Illegal character - "..."
The given invalid character was found in the input file.

Newline in middle of terminal name
The trailing comma was probably left off of the list of names.

Missing comma
A comma was missing.

Missing numeric value
The number was missing after a numeric capability.

NULL string value
The proper way to say that a string capability does not exist is to cancel it.

Very long string found. Missing comma?
Self-explanatory

Unknown option. Usage is:
An invalid option was entered.

Too many file names. Usage is:
Self-explanatory

"..." non-existent or permission denied
The given directory could not be written into.

"..." is not a directory
Self-explanatory

"...": Permission denied
Access denied.

"...": Not a directory
tic wanted to use the given name as a directory, but it already exists as a file.

SYSTEM ERROR!! Fork failed!!!
A **fork**(2) failed.

Error in following up use-links. Either there is a loop in the links or they reference non-existent terminals. The following is a list of the entries involved:

A **terminfo**(4) entry with a **use**=*name* capability either referenced a non-existant terminal called *name*, or *name* somehow referred back to the given entry.

WARNINGS

Total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes.

Terminal names exceeding 14 characters will be truncated to 14 characters and a warning message will be printed.

When the **–c** option is used, duplicate terminal names will not be diagnosed; however, when **–c** is not used, they will be.

BUGS

To allow existing executables from the previous release of the UNIX System to continue to run with the compiled terminfo entries created by the new terminfo compiler, canceled capabilities will not be marked as canceled within the terminfo binary unless the entry name has a '+' within it. (Such terminal names are only used for inclusion within other entries via a use= entry. Such names would not be used for real terminal names.)

For example:

```
4415+nl, kf1@, kf2@, ....

4415+base, kf1=\EOc, kf2=\EOd, ....

4415-nl|4415 terminal without keys,
   use=4415+nl, use=4415+base,
```

The above example works as expected; the definitions for the keys do not show up in the `4415-nl` entry. However, if the entry `4415+nl` did not have a plus sign within its name, the cancellations would not be marked within the compiled file and the definitions for the function keys would not be canceled within `4415-nl`.

NAME

time, ptime – time a command

SYNOPSIS

time [**–a**] [**–r**] [**–v**] command

ptime [**–a**] [**–r**] [**–v**] command

DESCRIPTION

The given command is executed; after it is complete, *time* prints the elapsed time during the command, the time spent in the system, and the time spent in execution of the command. Times are reported in seconds.

If the **–a** option is specified, or when invoked as *ptime, time* shows the execution times for each process.

If the **–r** option is specified, *time* also includes resource usage information.

The resource usage fields are:

pid	numerical user-id of process owner
ppid	numerical user-id of parent of process
MaxRSS	maximum real memory (resident set) size (in 1024 byte units)
MajorPF	major page faults
MinorPF	minor page faults
Swaps	count of process swaps
blkI	count of block input operations
blkO	count of block output operations
Nsig	count of signals received
Vcsw	voluntary context switches
Icsw	involuntary context switches

If the **–v** option is specified, *time* traces process events such as fork, exec, and exit as they occur.

The times are printed on the diagnostic output stream.

Time is a built-in command to *csh*(1), and *ksh*(1), with a different syntax and a different output format. This command is available as */bin/time* to *csh* and *ksh* users.

BUGS

Due to the multiprocessing nature of the operating system, it is possible for the CPU time (user + system time) for the command to greatly exceed the real time, thus giving percentages greater than 100% for CPU utilization.

The **–a** option returns a smaller number for the system time because it does not include the time used to exit the process.

NAME

timex – time a command; report process data and system activity

SYNOPSIS

timex [options] command

DESCRIPTION

The given *command* is executed and the elapsed time, user time, and system time spent in execution are reported in seconds. Optionally, process accounting data for the *command* and all of its children can be listed or summarized, and total system activity during the execution interval can be reported. The output of **timex** is written on standard error. The options for this command are as follows:

–p List process accounting records for *command* and all its children. Suboptions **f, h, k, m, r,** and **t** modify the data items reported. The options are as follows:

- **–f** Print the *fork/exec* flag and system exit status columns in the output.
- **–h** Instead of mean memory size, show the fraction of total available CPU time consumed by the process during its execution. This "hog factor" is computed as: (total CPU time)/(elapsed time).
- **–k** Instead of memory size, show total kcore-minutes.
- **–m** Show mean core size (the default).
- **–r** Show CPU factor (user time/(system-time + user time).
- **–t** Show separate system and user CPU times. The number of blocks read or written and the number of characters transferred are always reported.

–o Report the total number of blocks read or written and total characters transferred by *command* and all of its children.

–s Report total system activity (not just that due to *command*) that occurred during the execution interval of *command*. All data items listed in **sar**(1) are reported.

EXAMPLES

A simple example: `timex -ops sleep 60`

A terminal session of arbitrary complexity can be measured by timing a subshell:

```
timex -opskmt sh
        session commands
EOT
```

SEE ALSO

sar(1)

WARNING

Process records associated with *command* are selected from the accounting file */usr/adm/pacct* by inference, since process genealogy is not available. Background processes having the same user-id, terminal-id, and execution time window will be spuriously included.

NAME

toc: dtoc, ttoc, vtoc – graphical table of contents routines

SYNOPSIS

dtoc [directory]
ttoc mm-file
vtoc [**–cdhnimsvn**] [TTOC file]

DESCRIPTION

All of the commands listed below reside in */usr/bin/graf* (see **graphics**(1G)).

dtoc **dtoc** makes a textual table of contents, TTOC, of all subdirectories beginning at *directory* (*directory* defaults to .). The list has one entry per directory. The entry fields from left to right are level number, directory name, and the number of ordinary readable files in the directory. **dtoc** is useful in making a visual display of all or parts of a filesystem. The following will make a visual display of all the readable directories under /:

dtoc / | vtoc | td

ttoc Output is the table of contents generated by the .TC macro of **mm**(1) translated to TTOC format. The input is assumed to be an **mm** file that uses the .H family of macros for section headers (see the DOCUMENTER'S WORKBENCH Software). If no *file* is given, the standard input is assumed.

vtoc **vtoc** produces a GPS describing a hierarchy chart from a TTOC. The output drawing consists of boxes containing text connected in a tree structure. If no *file* is given, the standard input is assumed. Each TTOC entry describes one box and has the following form:

id [*line-weight*,*line-style*] "*text*" [*mark*]

id An alternating sequence of numbers and dots. The *id* specifies the position of the entry in the hierarchy. The *id* **0.** is the root of the tree.

line-weight Either:

n, normal-weight
m, medium-weight
b, bold-weight

line-style Either:

so, solid-line
do, dotted-line
dd, dot-dash line
da, dashed-liner
ld, long-dashed

text A character string surrounded by quotes. The characters between the quotes become the contents of the box. To include a quote within a box it must be escaped (\").

mark A character string (surrounded by quotes if it contains spaces), with included dots being escaped. The string is put above the top right corner of the box. To include either a quote or a dot within a **mark**, it must be

escaped.

Entry example: **1.1b,da "ABC" DEF**

Entries may span more than one line by escaping the newline (\newline).

Comments are surrounded by the /*,*/ pair. They may appear anywhere in a TTOC.

Options:

- **c** Use text as entered (default is all uppercase).
- **d** Connect the boxes with diagonal lines.
- **h***n* Horizontal interbox space is *n*% of box width.
- **i** Suppress the box *id*.
- **m** Suppress the box *mark*.
- **s** Do not compact boxes horizontally.
- **v***n* Vertical interbox space is *n*% of box height.

SEE ALSO

graphics(1G), gps(4)

mm(1) in the *DOCUMENTER'S WORKBENCH Technical Discussion and Reference Manual*

NAME

touch – update access and modification times of a file

SYNOPSIS

touch [**–amc**] [mmddhhmm[yy]] files

DESCRIPTION

touch causes the access and modification times of each argument to be updated. The filename is created if it does not exist. If no time is specified (see **date**(1)), the current time is used. The **–a** and **–m** options cause touch to update only the access or modification times respectively (default is **–am**). The **–c** option silently prevents **touch** from creating the file if it did not previously exist.

The return code from **touch** is the number of files for which the times could not be successfully modified (including files that did not exist and were not created).

SEE ALSO

date(1), utime(2)

NAME

tplot – graphics filters

SYNOPSIS

tplot [–Tterminal [–**e** raster]]

DESCRIPTION

These commands read plotting instructions (see **plot**(4)) from the standard input and in general produce, on the standard output, plotting instructions suitable for a particular *terminal*. If no *terminal* is specified, the environment parameter **$TERM** (see **environ**(5)) is used. Known *terminal*s are as follows:

300	DASI 300.
300S	DASI 300s.
450	DASI 450.
4014	Tektronix 4014.
ver	Versatec D1200A. This version of **plot** places a scan-converted image in */usr/tmp/raster*$$ and sends the result directly to the plotter device, rather than to the standard output. The –**e** option causes a previously scan-converted file *raster* to be sent to the plotter.

FILES

/usr/lib/t300
/usr/lib/t300s
/usr/lib/t450
/usr/lib/t4014
/usr/lib/vplot
/usr/tmp/raster$$

SEE ALSO

plot(3X), plot(4), term(5)

NAME

tput – initialize a terminal or query terminfo database

SYNOPSIS

tput [–Ttype] capname [parms ...]
tput [–Ttype] **init**
tput [–Ttype] **reset**
tput [–Ttype] **longname**
tput –S << *file*

DESCRIPTION

tput uses the **terminfo**(4) database to make the values of terminal-dependent capabilities and information available to the shell (see **sh**(1)), to initialize or reset the terminal, or to return the long name of the requested terminal type. **tput** outputs a string if the attribute (*cap*ability *name*) is of type string, or an integer if the attribute is of type integer. If the attribute is of type boolean, **tput** simply sets the exit code (**0** for TRUE if the terminal has the capability, **1** for FALSE if it does not), and produces no output. Before using a value returned on standard output, the user should test the exit code ($?, see **sh**(1)) to be sure it is **0**. (See **EXIT CODES** and **DIAGNOSTICS** below.) For a complete list of capabilities and the *capname* associated with each, see **terminfo**(4).

–T*type* Indicates the *type* of terminal. Normally this option is unnecessary, because the default is taken from the environment variable **TERM**. If **–T** is specified, then the shell variables **LINES** and **COLUMNS** will not be referenced.

capname Indicates the attribute from the **terminfo**(4) database.

parms If the attribute is a string that takes parameters, the arguments *parms* will be instantiated into the string. An all numeric argument will be passed to the attribute as a number.

–S Allows more than one capability per invocation of **tput**. The capabilities must be passed to **tput** from the standard input instead of from the command line (see example). Only one *capname* is allowed per line. The **–S** option changes the meaning of the **0** and **1** boolean and string exit codes (see **EXIT CODES**).

init If the **terminfo**(4) database is present and an entry for the user's terminal exists (see **–T***type*, above), the following will occur: (1) if present, the terminal's initialization strings will be output (**is1**, **is2**, **is3**, **if**, **iprog**), (2) any delays (e.g., newline) specified in the entry will be set in the tty driver, (3) tabs expansion will be turned on or off according to the specification in the entry, and (4) if tabs are not expanded, standard tabs will be set (every 8 spaces). If an entry does not contain the information needed for any of the above activities, that activity will silently be skipped.

reset Instead of putting out initialization strings, the terminal reset strings are output if present (**rs1**, **rs2**, **rs3**, **rf**). If the reset strings are not present but initialization strings are, the initialization strings are output. Otherwise, **reset** acts identically to **init**.

longname If the **terminfo**(4) database is present and an entry for the user's terminal exists (see **-T***type* above), then the long name of the terminal will be put out. The long name is the last name in the first line of the terminal's description in the **terminfo**(4) database (see **term**(5)).

EXAMPLES

tput init	Initialize the terminal according to the type of terminal in the environmental variable **TERM**. This command should be included in everyone's *.profile* after the environmental variable **TERM** has been exported, as illustrated on the **profile**(4) manual page.
tput -T5620 reset	Reset an AT&T 5620 terminal, overriding the type of terminal in the environmental variable **TERM**.
tput cup 0 0	Send the sequence to move the cursor to row **0**, column **0** (the upper-left corner of the screen, usually known as the "home" cursor position).
tput clear	Echo the clear-screen sequence for the current terminal.
tput cols	Print the number of columns for the current terminal.
tput -T450 cols	Print the number of columns for the 450 terminal.
bold=`tput smso`	
offbold=`tput rmso`	Set the shell variables **bold** (to begin stand-out mode sequence) and **offbold** (to end standout mode sequence) for the current terminal. This might be followed by a prompt:

```
echo "${bold}Please type in your name: ${offbold}\c"
```

tput hc	Set exit code to indicate if the current terminal is a hardcopy terminal.
tput cup 23 4	Send the sequence to move the cursor to row 23, column 4.
tput longname	Print the long name from the **terminfo**(4) database for the type of terminal specified in the environmental variable **TERM**.

```
tput -S <<!
> clear
> cup 10 10
> bold
> !
```

This example shows **tput** processing several capabilities in one invocation. This example clears the screen, moves the cursor to position 10, 10, and turns on bold (extra bright) mode. The list is terminated by an exclamation mark (!) on a line by itself.

FILES

/usr/lib/terminfo/?/*	Compiled terminal description database
/usr/include/curses.h	**curses**(3X) header file
/usr/include/term.h	**terminfo**(4) header file
/usr/lib/tabset/*	Tab settings for some terminals, in a format appropriate to be output to the terminal (escape

sequences that set margins and tabs); for more information, see the "Tabs and Initialization" section of **terminfo**(4).

SEE ALSO

stty (1), tabs (1), profile(4), terminfo(4)

DIAGNOSTICS

If *capname* is of type boolean, a value of **0** is set for TRUE and **1** for FALSE unless the **–S** option is used.

If *capname* is of type string, a value of **0** is set if the *capname* is defined for this terminal *type* (the value of *capname* is returned on standard output); a value of **1** is set if *capname* is not defined for this terminal *type* (a null value is returned on standard output).

If *capname* is of type boolean or string and the **–S** option is used, a value of **0** is returned to indicate that all lines were successful. No indication of which line failed can be given so exit code **1** will never appear. Exit codes **2**, **3**, and **4** retain their usual interpretation.

If *capname* is of type integer, a value of **0** is always set, whether or not *capname* is defined for this terminal *type*. To determine if *capname* is defined for this terminal *type*, the user must test the value of standard output. A value of **–1** means that *capname* is not defined for this terminal *type*.

Any other exit code indicates an error; see **DIAGNOSTICS**, below.

DIAGNOSTICS

tput prints these error messages and sets the corresponding exit codes:

Exit Code	Error Message
0	–1 (*capname* is a numeric variable not specified in the **terminfo**(4) database for this terminal type, e.g. **tput –T450 lines** and **tput –T2621 xmc**)
1	No error message is printed, see **EXIT CODES**, above
2	Usage error
3	Unknown terminal *type* or no **terminfo**(4) database
4	Unknown **terminfo**(4) capability *capname*

NAME

tr – translate characters

SYNOPSIS

tr [**–cds**] [string1 [string2]]

DESCRIPTION

tr copies the standard input to the standard output with substitution or deletion of selected characters. Input characters found in *string1* are mapped into the corresponding characters of *string2*. Any combination of the options **–cds** can be used:

–c Complements the set of characters in *string1* with respect to the universe of characters whose ASCII codes are 001 through 377 octal.

–d Deletes all input characters in *string1*.

–s Squeezes all strings of repeated output characters that are in *string2* to single characters.

The following abbreviation conventions can be used to introduce ranges of characters or repeated characters into the strings:

[a–z] Stands for the string of characters whose ASCII codes run from character **a** to character **z**, inclusive.

[a*n] Stands for *n* repetitions of **a**. If the first digit of *n* is **0**, *n* is considered octal; otherwise, *n* is taken to be decimal. A zero or missing *n* is taken to be huge; this facility is useful for padding *string2*.

The escape character (\) may be used as in the shell to remove special meaning from any character in a string. In addition, \ followed by 1, 2, or 3 octal digits stands for the character whose ASCII code is given by those digits.

EXAMPLES

The following example creates a list of all the words in *file1* one per line in *file2*, where a word is taken to be a maximal string of alphabetics. The strings are quoted to protect the special characters from interpretation by the shell; 012 is the ASCII code for newline.

```
tr –cs "[A–Z][a–z]" "[\012*]" <file1 >file2
```

SEE ALSO

csh(1), ed(1), ksh(1), sh(1), ascii(5)

BUGS

Will not handle ASCII NUL in *string1* or *string2*; always deletes NUL from input.

NAME

true, false – provide truth values

SYNOPSIS

true

false

DESCRIPTION

true does nothing, successfully. **false** does nothing, unsuccessfully. They are typically used in input to **sh**(1), as in the following example:

```
while true
do
        command
done
```

SEE ALSO

sh(1)

DIAGNOSTICS

true has exit status zero, **false** non-zero.

NAME

tsort – topological sort

SYNOPSIS

tsort [file]

DESCRIPTION

The **tsort** command produces on the standard output a totally ordered list of items consistent with a partial ordering of items mentioned in the input *file*. If no *file* is specified, the standard input is understood.

The input consists of pairs of items (nonempty strings) separated by blanks. Pairs of different items indicate ordering. Pairs of identical items indicate presence, but not ordering.

SEE ALSO

lorder(1)

DIAGNOSTICS

Odd data: there is an odd number of fields in the input file.

NAME

tty – get the name of the terminal

SYNOPSIS

tty [**–l**] [**–s**]

DESCRIPTION

tty prints the pathname of the user's terminal. The following options are available:

–l Prints the synchronous line number to which the user's terminal is connected, if it is on an active synchronous line.

–s Inhibits printing of the terminal pathname, allowing one to test just the exit code.

DIAGNOSTICS

not on an active synchronous line
The standard input is not a synchronous terminal and **–l** is specified.

not a tty The standard input is not a terminal and **–s** is not specified.

EXIT CODES:

2 If invalid options were specified

0 If standard input is a terminal

1 Otherwise

NAME

ttydevname – get the device associated with a terminal or the terminal associated with a device

SYNOPSIS

ttydevname <ttypath>

ttydevname <major> <minor>

DESCRIPTION

With one argument, **ttydevname** takes the basename of *ttypath* and prints the major and minor number of the device that would be returned by the **ttyname** library routine.

Because only the basename of *ttypath* is used, an obviously bad path may still print a valid major and minor number. For example, the path */dev/hello/world/ABCD* would use only ABCD, which is a valid basename.

To determine the major and minor device associated with your current terminal line, execute the following command:

ttydevname `tty`

With two arguments, **ttydevname** prints the path associated with *major* and *minor*.

DIAGNOSTICS

invalid name

The basename of *ttypath* is invalid. The basename must contain four characters; each character must be in the range [A-Za-z0-9_+].

SEE ALSO

tty(1), dev_to_ttyname(3C), ttyname(3C)

NAME

tunefs – tune up an existing filesystem

SYNOPSIS

/etc/tunefs *tuneup-options special | filesys*

DESCRIPTION

tunefs is designed to change the dynamic parameters of a filesystem which effect layout policies. The parameters are as follows:

–a *maxcontig*

The maximum number of contiguous blocks that are laid out before forcing a rotational delay (see **–d**). The default value is one, since most device drivers require an interrupt per disk transfer. Device drivers that can chain several buffers together in a single transfer should set this to the maximum chain length.

–d *rotdelay*

The expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file.

–e *maxbpg*

The maximum number of blocks a one file that can be allocated out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically, this is one-fourth of the number of blocks in a cylinder group. This prevents all of the blocks in a single cylinder group from being allocated to one file, which degrades the access time for all other files allocated to that cylinder group. This increases the number of long seeks on large files. For filesystems with large files exclusively, *maxbpg* should be set higher.

–m *minfree*

The percentage of space withheld from normal users; the minimum free-space threshold. The default value is 10%. This value can be set to zero; however, throughput at a level of zero is approximately one-third of the throughput at a 10% threshold. Note that if the value is raised above the current usage level, users will be unable to allocate files until enough files have been deleted to get under the higher threshold.

SEE ALSO

newfs(1M), mkfs(1M), fs(4)

BUGS

tunefs should work on mounted and active filesystems. Because the superblock is not kept in the buffer cache, the changes take effect only if **tunefs** is run on dismounted filesystems. If **tunefs** is run on the root filesystem, the system must be rebooted.

NAME

uadmin – administrative control

SYNOPSIS

/etc/uadmin cmd fcn [mdep]

DESCRIPTION

The **uadmin** command provides control for basic administrative functions. This command is tightly coupled to the System Administration procedures and is not intended for general use. Use of this command may produce unexpected results. It may be invoked only by the superuser.

The arguments *cmd* (command), *fcn* (function) and, optionally, *mdep* (machine-dependent argument) are converted to integers and passed as the first, second and third arguments to the **uadmin** system call. All online processors are off-lined before making the call.

For example, *uadmin 2 0* will kill all processes, unmount the root filesystem and return to firmware. *uadmin 1 4 4* will return to firmware immediately, without flushing the buffer cache out to disk, or killing all processes, or unmounting the root filesystem.

SEE ALSO

uadmin(2)

WARNINGS

Only one **uadmin** command at a time may run on the operating system. **uadmin** commands are found in the *inittab* file.

If a **uadmin** command is present in an *rc* file, the error message `uadmin: invalid argument` may be issued. This is because the operating system may be processing one **uadmin** command when the next is encountered, causing execution of the commands to overlap. If any of the *rc* files contain **uadmin** commands, make sure that the *inittab* file does not also contain a **uadmin** command that executes at the same run level.

NAME

umask – set file-creation mode mask

SYNOPSIS

umask [ooo]

DESCRIPTION

The user file-creation mode mask is set to *ooo*. The three octal digits refer to read/write/execute permissions for *owner*, *group*, and *others*, respectively (see **chmod**(2) and **umask**(2)). The value of each specified digit is subtracted from the corresponding "digit" specified by the system for the creation of a file (see **creat**(2)). For example, **umask 022** removes *group* and *others* write permission (files normally created with mode **777** become mode **755**; files created with mode **666** become mode **644**).

If *ooo* is omitted, the current value of the mask is printed.

umask is recognized and executed by the shell.

For Bourne and Korn shell users, **umask** can be included in the user's *.profile* (see **profile**(4)) and invoked at login to automatically set the user's permissions on files or directories created. For C shell users, **umask** can be included in the user's *.login* file.

SEE ALSO

chmod(1), csh(1), ksh(1), sh(1), chmod(2), creat(2), umask(2), profile(4)

NAME

uname – print name of current system

SYNOPSIS

uname [**–snrvma**]
uname [**–S** system name]

DESCRIPTION

uname prints the current system name on the standard output file. It is mainly useful to determine which system one is using. The options cause selected information returned by **uname**(2) to print:

–s Print the system name (default).

–n Print the nodename (the nodename is the name by which the system is known to a communications network).

–r Print the operating system release.

–v Print the operating system version.

–m Print the machine hardware name.

–a Print all of the above information.

The system name and the nodename may be changed by specifying a system name argument to the **–S** option. The system name argument is restricted to 8 characters. Only the superuser is allowed this capability.

SEE ALSO

uname(2)

NAME

unget – undo a previous get of an SCCS file

SYNOPSIS

unget [–rSID] [–**s**] [–**n**] files

DESCRIPTION

unget undoes the effect of a **get** –**e** done prior to creating the intended new delta. If a directory is named, **unget** behaves as though each file in the directory were specified as a named file, except that non-SCCS files and unreadable files are silently ignored. If a name of – is given, the standard input is read with each line being taken as the name of an SCCS file to be processed.

Keyletter arguments apply independently to each named file.

–r*SID* Uniquely identifies which delta is no longer intended. (This would have been specified by **get** as the "new delta"). The use of this keyletter is necessary only if two or more outstanding **get**s for editing on the same SCCS file were done by the same person (login name). A diagnostic results if the specified *SID* is ambiguous, or if it is necessary and omitted on the command line.

–**s** Suppresses the printout, on the standard output, of the intended delta's *SID*.

–**n** Causes the retention of the gotten file which would normally be removed from the current directory.

SEE ALSO

delta(1), get(1), help(1), sact(1)

DIAGNOSTICS

Use **help**(1) for explanations.

NAME

uniq – report repeated lines in a file

SYNOPSIS

uniq [**–udc** [+n] [–n]] [input [output]]

DESCRIPTION

uniq reads the input file and compares adjacent lines. In the normal case, the second and succeeding copies of repeated lines are removed; the remainder is written on the output file. *input* and *output* should always be different. Note that repeated lines must be adjacent in order to be found; see **sort**(1). If the **–u** flag is used, just the lines that are not repeated in the original file are output. The **–d** option specifies that one copy of just the repeated lines is to be written. The normal mode output is the union of the **–u** and **–d** mode outputs.

The **–c** option supersedes **–u** and **–d** and generates an output report in default style but with each line preceded by a count of the number of times it occurred.

The *n* arguments specify skipping an initial portion of each line in the comparison:

–n The first *n* fields together with any blanks before each are ignored. A field is defined as a string of non-space, non-tab characters separated by tabs and spaces from its neighbors.

+n The first *n* characters are ignored. Fields are skipped before characters.

SEE ALSO

comm(1), sort(1)

NAME

units – conversion program

SYNOPSIS

units

DESCRIPTION

units converts quantities expressed in various standard scales to their equivalents in other scales. It works interactively in this fashion:

```
You have: inch
You want: cm
        * 2.540000e+00
        / 3.937008e-01
```

A quantity is specified as a multiplicative combination of units optionally preceded by a numeric multiplier. Powers are indicated by suffixed positive integers, division by the usual sign.

```
You have: 15 lbs force/in2
You want: atm
        * 1.020689e+00
        / 9.797299e-01
```

units only does multiplicative scale changes; thus it can convert Kelvin to Rankine, but not Celsius to Fahrenheit. Most familiar units, abbreviations, and metric prefixes are recognized, together with a generous leavening of exotica and a few constants of nature including the following:

pi	Ratio of circumference to diameter
c	Speed of light
e	Charge on an electron
g	Acceleration of gravity
force	Same as **g**
mole	Avogadro's number
water	Pressure head per unit height of water
au	Astronomical unit

Pound is not recognized as a unit of mass; **lb** is. Compound names are run together, (e.g., **lightyear**). British units that differ from their U.S. counterparts are prefixed thus: **brgallon**. For a complete list of units, type this command:

cat /usr/lib/unittab

FILES

/usr/lib/unittab

NAME

uptime – show how long system has been up

SYNOPSIS

uptime

DESCRIPTION

uptime prints the current time, the length of time the system has been up, and the average number of jobs in the run queue over the last 1, 5 and 15 minutes.

FILES

/unix system name list

NAME

usage – retrieve a command description and usage examples

SYNOPSIS

[**help**] **usage** [**–d**] [**–e**] [**–o**] [command_name]

DESCRIPTION

The Help Facility **usage** command retrieves information about operating system commands. With no argument, **usage** displays a menu that prompts the user for the name of a command, or allows the user to retrieve a list of commands supported by **usage**. The user may also exit to the shell by typing q (for "quit").

After a command is selected, the user is asked to choose either a description of the command, examples of typical usage of the command, or descriptions of the command's options. Then, based on the user's request, the appropriate information will be printed.

A command name may also be entered at shell level as an argument to **usage**. To receive information on the command's description, examples, or options, the user can use the **–d**, **–e**, or **–o** options respectively. (The default option is **–d**.)

From any screen in the Help Facility, a user can execute a command via the shell by typing an exclamation mark (!) and the command. The screen will be redrawn if the command was entered at a first-level prompt. If the command is entered at any other prompt level, only the prompt will be redrawn.

By default, the Help Facility scrolls the data that is presented to the user. If you prefer to have the screen clear before printing the data (non-scrolling), the shell variable **SCROLL** must be set to **no** and exported so it will become part of your environment. This is done by adding the following line to your *.profile* file (see **profile**(4)): "`export SCROLL ; SCROLL=no`". If you later decide that scrolling is desired, **SCROLL** must be set to **yes**.

Information on each of the Help Facility commands (**starter**, **locate**, **usage**, **glossary**, and **help**) is located on their respective manual pages.

SEE ALSO

csh(1), glossary(1), help(1), ksh(1), locate(1), sh(1), starter(1), term(5)

WARNINGS

If the shell variable **TERM** (see **sh**(1)) is not set in the user's *.profile* file, then **TERM** will default to the terminal value type 450 (a hard-copy terminal). For a list of valid terminal types, refer to **term**(5).

NAME

uucheck – check the uucp directories and permissions file

SYNOPSIS

/usr/lib/uucp/uucheck [**–v**] [**–x** debug_level]

DESCRIPTION

uucheck checks for the presence of the **uucp** system required files and directories. Within the **uucp** makefile, it is executed before the installation takes place. It also checks for some obvious errors in the *Permissions* file (*/usr/lib/uucp/Permissions*). When executed with the **–v** option, it gives a detailed explanation of how the **uucp** programs will interpret the *Permissions* file. The **–x** option is used for debugging. *debug-option* is a single digit in the range 1-9; the higher the value, the greater the detail.

Note that **uucheck** can be used only by the superuser or **uucp**.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Maxuuscheds
/usr/lib/uucp/Maxuuxqts
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO

uucp(1C), uustat(1C), uux(1C), uucico(1M), uusched(1M)

BUGS

The program does not check file/directory modes or some errors in the *Permissions* file such as duplicate login or machine name.

NAME

uucico – file transport program for the uucp system

SYNOPSIS

/usr/lib/uucp/uucico [–r role_number] [–x debug_level]
[–i interface] [–d spool_directory] –s system_name

DESCRIPTION

uucico is the file transport program for **uucp** work file transfers. Role numbers for the **–r** option are the digit 1 for master mode or 0 for slave mode (default). The **–r** option should be specified as the digit 1 for master mode when **uucico** is started by a program or **cron. uux** and **uucp** both queue jobs that will be transferred by **uucico.** It is normally started by the scheduler, **uusched,** but can be started manually; this is done for debugging. For example, the shell **Uutry** starts **uucico** with debugging turned on. A single digit must be used for the **–x** option with higher numbers for more debugging.

The **–i** option defines the **interface** used with **uucico.** This interface only affects slave mode. Known interfaces are UNIX (default), TLI (basic Transport Layer Interface), and TLIS (Transport Layer Interface with Streams modules, read/write).

The **–d** option is used to specify the directory (*spool_directory*) containing the work files to be transferred. The default spool directory is */usr/spool/uucp*. The **–s** option defines the system (*system_name*) that **uucico** will try to contact. The *system_name* must be defined in the *Systems* file.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Devconfig
/usr/lib/uucp/Sysfiles
/usr/lib/uucp/Maxuuxqts
/usr/lib/uucp/Maxuuscheds
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO

cron(1M), uucp(1C), uusched(1M), uustat(1C), uutry(1M), uux(1C)

NAME

uucleanup – uucp spool directory clean-up

SYNOPSIS

/usr/lib/uucp/uucleanup [**–C***time*] [**–W***time*] [**–X***time*] [**–m***string*] [**–o***time*] [**–s***system*]

DESCRIPTION

uucleanup will scan the spool directories for old files and take appropriate action to remove them in a useful way:

Inform the requestor of send/receive requests for systems that cannot be reached.

Return mail that cannot be delivered to the sender.

Delete or execute rnews for rnews type files (depending on where the news originated—locally or remotely).

Remove all other files.

In addition, there is provision to warn users of requests that have been waiting for a given number of days (default 1). Note that **uucleanup** will process as if all option *times* were specified to the default values unless *time* is specifically set.

The following options are available.

–C*time* Any *C.* files greater or equal to *time* days old will be removed with appropriate information to the requestor. (Default 7 days.)

–D*time* Any *D.* files greater or equal to *time* days old will be removed. An attempt will be made to deliver mail messages and execute rnews when appropriate. (Default 7 days.)

–W*time* Any *C.* files equal to *time* days old will cause a mail message to be sent to the requestor warning about the delay in contacting the remote. The message includes the *JOBID*, and in the case of mail, the mail message. The administrator may include a message line telling who to call to check the problem (**–m** option). (Default 1 day.)

–X*time* Any *X.* files greater or equal to *time* days old will be removed. The *D.* files are probably not present (if they were, the **2.** could get executed). But if there are *D.* files, they will be taken care of by D. processing. (Default 2 days.)

–m*string* This line will be included in the warning message generated by the **–W** option.

–o*time* Other files whose age is more than *time* days will be deleted. (Default 2 days.) The default line is `See your local administrator to locate the problem.`

–s*system* Execute for *system* spool directory only.

–x*debug_level*
The **–x** debug level is a single digit between 0 and 9; higher numbers give more detailed debugging information. (If **uucleanup** was compiled with -DSMALL, no debugging output will be available.)

This program is typically started by the shell **uudemon.cleanup**, which should be started by **cron**(1M).

FILES

/usr/lib/uucp	Directory with commands used by **uucleanup** internally
/usr/spool/uucp	Spool directory

SEE ALSO

uucp(1C), uux(1C), cron(1M)

NAME

uucp, uulog, uuname – UNIX-to-UNIX system copy

SYNOPSIS

uucp [options] source-files destination-file
uulog [options] **–s**system
uulog [options] system
uulog [options] **–f**system
uuname [**–l**] [**–c**]

DESCRIPTION

uucp

uucp copies files named by the *source-file* arguments to the *destination-file* argument. A filename can be a pathname on your machine, or can have this form:

system-name!*pathname*

system-name is taken from a list of system names that **uucp** knows about. The *system-name* can also be a list of names such as the following:

system-name!*system-name*!...!*system-name*!*pathname*

In this case, an attempt is made to send the file to the destination via the specified route. See **WARNINGS** and **BUGS** below for restrictions. Care should be taken to ensure that intermediate nodes in the route are willing to foward information (see **WARNINGS** below for restrictions).

The following shell metacharacters are disallowed in *system-name*:

` ; & | ^ < > ( ) <CR> <TAB> <SPACE>

Pathnames may be one of the following:

(1) A full pathname.

(2) A pathname preceded by ~*user* where *user* is a login name on the specified system and is replaced by that user's login directory.

(3) A pathname preceded by ~*/destination* where *destination* is appended to */usr/spool/uucppublic*.

(NOTE: This destination will be treated as a filename unless more than one file is being transfered by this request or the destination is already a directory. To ensure that it is a directory, follow the destination with a slash (/). For example ~**/dan/** as the destination will create the directory */usr/spool/uucppublic/dan* if it does not exist and put the requested file(s) in that directory).

(4) Anything else is prefixed by the current directory.

If the result is an erroneous pathname for the remote system, the copy will fail. If the *destination-file* is a directory, the last part of the *source-file* name is used.

uucp preserves execute permissions across the transmission and gives 0666 read and write permissions (see **chmod**(2)).

The following options are interpreted by **uucp**:

–c Do not copy local file to the spool directory for transfer to the remote machine (default).

–C Force the copy of local files to the spool directory for transfer.

–d Make all necessary directories for the file copy (default).

–f Do not make intermediate directories for the file copy.

–g*grade* *Grade* is a single letter/number; lower ASCII sequence characters will cause the job to be transmitted earlier during a particular conversation.

–j Output the job identification ASCII string on the standard output. This job identification can be used by **uustat** to obtain the status or terminate a job.

–m Send mail to the requester when the copy is completed.

–n*user* Notify *user* on the remote system that a file was sent.

–r Do not start the file transfer, just queue the job.

–s*file* Report status of the transfer to *file*. Note that the *file* must be a full pathname.

–x*debug_level*
Produce debugging output on standard output. The *debug_level* is a number between 0 and 9; higher numbers give more detailed information. (Debugging will not be available if **uucp** was compiled with **-DSMALL**.)

uulog:

uulog queries */usr/spool/uucp/.Log/uucico/system* (the **uucp** transactions log file) and */usr/spool/uucp/.Log/uuxqt/system* (the **uuxqt** transactions log file).

The options cause **uulog** to print logging information:

–s*sys* Print information about file transfer work involving system *sys*.

–f*system* Does a **tail –f** of the file transfer log for *system*. (You must press Break to exit this function.) Other options used in conjunction with the above:

–x Look in the **uuxqt** log file for the given system.

–*number* Indicates that a **tail** command of *number* lines should be executed.

uuname:

uuname lists the names of systems known to **uucp**. The **–c** option returns the names of systems known to **cu**. (The two lists are the same, unless your machine is using different *Systems* files for **cu** and **uucp**. See the *Sysfiles* file.) The **–l** option returns the local system name.

FILES

/usr/spool/uucp	Spool directories
/usr/spool/uucppublic/*	Public directory for receiving and sending (*/usr/spool/uucppublic*)
/usr/lib/uucp/*	Other data and program files

SEE ALSO

mail(1), uustat(1C), uux(1C), uuxqt(1M), chmod(2)

WARNINGS

The domain of remotely accessible files can (and for obvious security reasons, usually should) be severely restricted. You will very likely not be able to fetch files by pathname; ask a responsible person on the remote system to send them to you. For the same reasons, you will probably not be able to send files to arbitrary pathnames. As distributed, the remotely accessible files are those whose names begin */usr/spool/uucppublic* (equivalent to *~/*).

All files received by **uucp** will be owned by **uucp**.

The **–m** option works only to send files or receive a single file. Receiving multiple files specified by special shell characters **? * [...]** will not activate the **–m** option.

The forwarding of files through other systems may not be compatible with the previous version of **uucp**. If forwarding is used, all systems in the route must have the same version of **uucp**.

BUGS

Protected files and files that are in protected directories owned by the requestor can be sent via **uucp** by using the **–C** option. However, if the requestor is **root** and the directory is not searchable by "other" or the file is not readable by "other," the request will fail.

NAME

uugetty – set terminal type, modes, speed, and line discipline

SYNOPSIS

/usr/lib/uucp/uugetty [**–t** *timeout*] [**–r**] *line* [*speed* [*type* [*linedisc*]]]
/usr/lib/uucp/uugetty –c *file*

DESCRIPTION

uugetty is identical to **getty**(1M), but changes have been made to support using the line for **uucico, cu,** and **ct**; that is, the line can be used in both directions. The **uugetty** will allow users to login, but if the line is free, **uucico, cu,** or **ct** can use it for dialing out. The implementation depends on the fact that **uucico, cu,** and **ct** create lock files when devices are used. When the "open()" returns (or the first character is read when the **–r** option is used), the status of the lock file indicates whether the line is being used by **uucico, cu, ct,** or someone trying to login. Note that in the **–r** case, several <carriage-return> characters may be required before the login message is output. The human users will be able to handle this slight inconvenience. **uucico** trying to log in will have to be told by using the following login script:

"" \r\d\r\d\r\d\r in:--in: . . .

The . . . is whatever would normally be used for the login sequence.

If there is a **uugetty** on one end of a direct line, there must be a **uugetty** on the other end as well. Here is an */etc/inittab* entry using **uugetty** on an intelligent modem or direct line:

```
30:2:respawn:/usr/lib/uucp/uugetty -r -t 60 ttyGA/GAAH 1200
```

The meanings of the available options are as follows:

–t *timeout* — Specifies that **uugetty** should exit if the open on the line succeeds and there is no response to the login prompt in *timeout* seconds. *timeout* is replaced by an integer.

–r — Causes **uugetty** to wait to read a character before it puts out the login message, thus preventing two **uugettys** from looping. An entry for an intelligent modem or direct line that has a **uugetty** on each end must use this option.

line — Defines the name of the line to which **uugetty** will attach itself. The line name will point to an entry in the */dev* directory. For example, */dev/ttyGA/GAAB*.

speed — Defines the entry to use from the */etc/gettydefs* file. The entry defines the line speed, the login message, the initial tty setting, and the next speed to try if the user says the speed is inappropriate (by sending a *break* character). The default *speed* is 300.

type — Defines the type of terminal connected to the line. The default terminal is **none**, representing a normal terminal unknown to the system (see **getty**(1M)).

linedisc — Sets the line discipline to use on the line. The default is line discipline 0, which is the only one currently compiled into the operating system.

-c *file* Checks the speed and tty definitions in *file* and sends the results to standard output. Unrecognized modes and improperly constructed entries are reported. For correct entries, flag values are printed. *file* is replaced by */etc/gettydefs* or a similarly structured file.

FILES

/etc/gettydefs
/etc/issue

SEE ALSO

ct(1C), cu(1C), getty(1M), init(1M), login(1) uucico(1M), ioctl(2), gettydefs(4), inittab(4), tty(7)

BUGS

ct will not work when **uugetty** is used with an intelligent modem such as penril or ventel.

NAME

uusched – the scheduler for the uucp file transport program

SYNOPSIS

/usr/lib/uucp/uusched [**–x** debug_level] [**–u** debug_level]

DESCRIPTION

uusched is the **uucp** file transport scheduler. It is usually started by the daemon **uudemon.hour** that is started by **cron**(1M) from an entry in */usr/spool/cron/crontab*:

```
39 * * * * /bin/su uucp -c "/usr/lib/uucp/uudemon.hour >
/dev/null"
```

The two options are for debugging purposes only: **–x** *debug_level* will output debugging messages from **uusched** and **–u** *debug_level* will be passed as **–x** *debug_level* to **uucico**. The *debug_level* is a number between 0 and 9; higher numbers give more detailed information.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO

uucp(1C), uustat(1C), uux(1C), cron(1M), uucico(1M)

NAME

uustat – uucp status inquiry and job control

SYNOPSIS

uustat [–a]
uustat [–m]
uustat [–p]
uustat [–q]
uustat [–k*jobid*]
uustat [–r*jobid*]
uustat [–s*system*] [–u*user*]

DESCRIPTION

uustat will display the status of previously specified **uucp** commands, cancel previous **uucp** commands, or provide general status on **uucp** connections to other systems. Only one of the following options can be specified with **uustat** per command execution:

–a Output all jobs in queue.

–m Report the status of accessibility of all machines.

–p Execute a **ps –flp** for all the process-ids that are in the lock files.

–q List the jobs queued for each machine. If a status file exists for the machine, its date, time and status information are reported. In addition, if a number appears in () next to the number of *C* or *X* files, it is the age in days of the oldest *C./X.* file for that system. The Retry field represents the number of hours until the next possible call. The Count is the number of failure attempts. NOTE: For systems with a moderate number of outstanding jobs, this could take 30 seconds or more of real-time to execute. The following example shows output produced by the **–q** option:

```
eagle       3C      04/07-11:07     NO DEVICES AVAILABLE
mh3bs3      2C      07/07-10:42     SUCCESSFUL
```

The above output tells how many command files are waiting for each system. Each command file may have zero or more files to be sent (zero means to call the system and see if work is to be done). The date and time refer to the previous interaction with the system followed by the status of the interaction.

–k*jobid* Kill the **uucp** request whose job identification is *jobid*. The killed **uucp** request must belong to the person issuing the **uustat** command unless one is the superuser.

–r*jobid* Rejuvenate *jobid*. The files associated with *jobid* are touched so that their modification time is set to the current time. This prevents the cleanup daemon from deleting the job until the jobs modification time reaches the limit imposed by the deamon.

Either or both of the following options can be specified with **uustat:**

–s*sys* Report the status of all **uucp** requests for remote system *sys*.

–u*user* Report the status of all **uucp** requests issued by *user*.

Output for both the **-s** and **-u** options has the following format:

```
eaglen0000   4/07-11:01:03    (POLL)
eagleN1bd7   4/07-11:07       Seagledan522 /usr/dan/A
eagleC1bd8   4/07-11:07       Seagledan59 D.3b2al2ce4924
                 4/07-11:07   Seagledanrmail mike
```

With the above two options, the first field is the *jobid* of the job. This is followed by the date/time. The next field is either an S or an R depending on whether the job is to send or request a file. This is followed by the user-id of the user who queued the job. The next field contains the size of the file, or in the case of a remote execution (**rmail**—the command used for remote mail), the name of the command. When the size appears in this field, the filename is also given. This can be either the name given by the user or an internal name (such as `D.3b2alce4924`) that is created for data files associated with remote executions (**rmail** in this example).

When no options are given, **uustat** outputs the status of all **uucp** requests issued by the current user.

FILES

/usr/spool/uucp/* Spool directories

SEE ALSO

uucp(1C)

NAME

uuto, uupick – public UNIX-to-UNIX system file copy

SYNOPSIS

uuto [options] source-files destination
uupick [**–s** system]

DESCRIPTION

uuto sends *source-files* to *destination*. **uuto** uses the **uucp**(1C) facility to send files, while it allows the local system to control the file access. A source-file name is a pathname on your machine. Destination has the form:

system!*user*

system is taken from a list of system names that **uucp** knows about (see **uuname**). *user* is the login name of someone on the specified system.

Two options are available:

–p Copy the source file into the spool directory before transmission.

–m Send mail to the sender when the copy is complete.

The files (or sub-trees if directories are specified) are sent to PUBDIR on *system*, where PUBDIR is a public directory defined in the **uucp** source. By default this directory is */usr/spool/uucppublic*. Specifically the files are sent to:

PUBDIR/*receive*/*user*/*mysystem*/*files*

The destined recipient is notified by **mail**(1) of the arrival of files.

uupick accepts or rejects the files transmitted to the user. Specifically, **uupick** searches PUBDIR for files destined for the user. For each entry (file or directory) found, the following message is printed on the standard output:

from *system* : [file *file-name*] [dir *dirname*] ?

uupick then reads a line from the standard input to determine the disposition of the file:

newline	Go to next entry.
d	Delete the entry.
m [*dir*]	Move the entry to named directory *dir*. If *dir* is not specified as a complete pathname (in which **$HOME** is legitimate), a destination relative to the current directory is assumed. If no destination is given, the default is the current directory.
a [*dir*]	Same as **m**, except moves all files sent from *system*.
p	Print the content of the file.
q	Stop.
EOT (Ctrl-D)	Same as **q**.
!*command*	Escape to the shell to do *command*.
*	Print a command summary.

uupick invoked with the **–s***system* option will search only the PUBDIR for files sent from *system*.

FILES

PUBDIR	/usr/spool/uucppublic	public directory

SEE ALSO

mail(1), uucp(1C), uustat(1C), uux(1C), uucleanup(1M)

WARNINGS

To send files that begin with a dot (such as *.profile*) the files must by qualified with a dot. For example: *.profile*, *.prof**, *.profil?* are correct; whereas **prof**, *?profile* are incorrect.

NAME

Uutry – try to contact remote system with debugging on

SYNOPSIS

/usr/lib/uucp/Uutry [**–x** debug_level] [**–r**] system_name

DESCRIPTION

Uutry is a shell that is used to invoke **uucico** to call a remote site. Debugging is turned on (default is level 5); **–x** will override that value. The **–r** overrides the retry time in */usr/spool/uucp/.status*. The debugging output is put in file */tmp/system_name*. A tail **–f** of the output is executed. A Delete or Break character will give control back to the terminal while the **uucico** continues to run, putting its output in */tmp/system_name*.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Maxuuxqts
/usr/lib/uucp/Maxuuscheds
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*
/tmp/system_name

SEE ALSO

uucp(1C), uux(1C), uucico(1M)

NAME

uux – UNIX-to-UNIX system command execution

SYNOPSIS

uux [options] command-string

DESCRIPTION

uux will gather zero or more files from various systems, execute a command on a specified system, and then send standard output to a file on a specified system.

NOTE: For security reasons, most installations limit the list of commands executable on behalf of an incoming request from **uux**, permiting only the receipt of mail (see **mail**(1)). (Remote execution permissions are defined in */usr/lib/uucp/Permissions*.)

The *command-string* is made up of one or more arguments that look like a shell command line, except the command and filenames can be prefixed by *system-name*!. A null *system-name* is interpreted as the local system.

Filenames may be one of the following:

(1) A full pathname

(2) A pathname preceded by ~*xxx* where *xxx* is a login name on the specified system and is replaced by that user's login directory.

(3) Anything else is prefixed by the current directory.

As an example, the following command gets *file1* from the *usg* machine and *file2* from the *pwba* machine, executes a **diff**(1) command, and put the results in *file.diff* in the local *PUBDIR/dan/* directory.

uux "!diff usg!/usr/dan/file1 pwba!/a4/dan/file2 > ~/dan/file.diff"

Any special shell characters such as <>;| should be quoted either by quoting the entire *command-string* or by quoting the special characters as individual arguments.

uux will attempt to get all files to the execution system. For files that are output files, the filename must be escaped by using parentheses. For example, the following command gets */usr/file* from system *b* and sends it to system *a*, performs a **cut** command on that file, and sends the result of the **cut** command to system *c*.

uux a!cut -f1 b!/usr/file \(c!/usr/file\)

uux will notify you if the requested command on the remote system was disallowed. This notification can be turned off by the **-n** option. The response comes by remote mail from the remote machine.

The following *options* are interpreted by **uux**:

- The standard input to **uux** is made the standard input to the *command-string*.

-a*name* Use *name* as the user identification replacing the initiator user-id. (Notification will be returned to the user.)

-b Return whatever standard input was provided to the **uux** command if the exit status is non-zero.

–c Do not copy local file to the spool directory for transfer to the remote machine (default).

–C Force the copy of local files to the spool directory for transfer.

–g*grade* *Grade* is a single letter/number; lower ASCII sequence characters will cause the job to be transmitted earlier during a particular conversation.

–j Output the *jobid* ASCII string on the standard output which is the job identification. This job identification can be used by **uustat** to obtain the status of a job or to terminate it.

–n Do not notify the user if the command fails.

–p Same as –. The standard input to **uux** is made the standard input to the *command-string*.

–r Do not start the file transfer, just queue the job.

–s*file* Report status of the transfer in *file*.

–x*debug_level*
Produce debugging output on the standard output. The *debug_level* is a number between 0 and 9; higher numbers give more detailed information.

–z Send success notification to the user.

FILES

/usr/lib/uucp/spool	Spool directories
/usr/lib/uucp/Permissions	Remote execution permissions
/usr/lib/uucp/*	Other data and programs

SEE ALSO

cut(1), mail(1), uucp(1C), uustat(1C)

WARNINGS

Only the first command of a shell pipeline may have a *system-name*!. All other commands are executed on the system of the first command.

The use of the shell metacharacter * will probably not do what you want it to do. The shell tokens << and >> are not implemented.

The execution of commands on remote systems takes place in an execution directory known to the **uucp** system. All files required for the execution will be put into this directory unless they already reside on that machine. Therefore, the simple filename (without path or machine reference) must be unique within the **uux** request. The following command will not work:

uux "a!diff b!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"

The next command will work if **diff** is a permitted command.

uux "a!diff a!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"

BUGS

Protected files and files that are in protected directories that are owned by the requestor can be sent in commands using **uux**. However, if the requestor is **root**, and the directory is not searchable by "other," the request will fail.

NAME

uuxqt – execute remote command requests

SYNOPSIS

/usr/lib/uucp/uuxqt [**–s** system] [**–x** debug_level]

DESCRIPTION

uuxqt is a program that executes remote job requests from remote systems generated by the use of the **uux** command. (**mail** uses **uux** for remote mail requests). **uuxqt** searches the spool directories looking for *X.* files. For each *X.* file, **uuxqt** checks to see if all the required data files are available and accessible and if file commands are permitted for the requesting system. The *Permissions* file is used to validate file accessibility and command execution permission.

There are two environment variables that are set before the **uuxqt** command is executed:

UU_MACHINE is the machine that sent the job (the previous one).

UU_USER is the user that sent the job.

These can be used in writing commands that remote systems can execute to provide information, auditing, or restrictions.

The **-x** *debug_level* is a single digit between 0 and 9. Higher numbers give more detailed debugging information.

FILES

/usr/lib/uucp/Permissions
/usr/lib/uucp/Maxuuxqts
/usr/spool/uucp/*
/usr/spool/locks/LCK*

SEE ALSO

uucp(1C), uustat(1C), uux(1C), mail(1), uucico(1M)

NAME

val – validate SCCS file

SYNOPSIS

val –
val [**–s**] [**–r**SID] [**–m**name] [**–y**type] files

DESCRIPTION

val determines if the specified *file* is an SCCS file meeting the characteristics specified by the optional argument list. Arguments to **val** may appear in any order. The arguments consist of keyletter arguments, which begin with a –, and named files.

val has a special argument, –, which causes reading of the standard input until an end-of-file condition is detected. Each line read is independently processed as if it were a command line argument list.

val generates diagnostic messages on the standard output for each command line and file processed, and also returns a single 8-bit code upon exit as described below.

The keyletter arguments are defined as follows. The effects of any keyletter argument apply independently to each named file on the command line.

–s The presence of this argument silences the diagnostic message normally generated on the standard output for any error that is detected while processing each named file on a given command line.

–r*SID* The argument value *SID* (SCCS Identification String) is an SCCS delta number. A check is made to determine if the *SID* is ambiguous (for example, **r**1 is ambiguous because it physically does not exist but implies 1.1, 1.2, etc., which may exist) or invalid (for example, **r**1.0 or **r**1.1.0 are invalid because neither case can exist as a valid delta number). If the *SID* is valid and not ambiguous, a check is made to determine if it actually exists.

–mname The argument value *name* is compared with the SCCS %M% keyword in *file*.

–ytype The argument value *type* is compared with the SCCS %Y% keyword in *file*.

The 8-bit code returned by **val** is a disjunction of the possible errors. That is, it can be interpreted as a bit string where (moving from left to right) set bits are interpreted as follows:

bit 0 = missing file argument
bit 1 = unknown or duplicate keyletter argument
bit 2 = corrupted SCCS file
bit 3 = cannot open file or file not SCCS
bit 4 = *SID* is invalid or ambiguous
bit 5 = *SID* does not exist
bit 6 = %Y%, **–y** mismatch
bit 7 = %M%, **–m** mismatch

Note that **val** can process two or more files on a given command line and in turn can process multiple command lines (when reading the standard input). In these cases, an aggregate code is returned—a logical **OR** of the codes

generated for each command line and file processed.

SEE ALSO

admin(1), delta(1), get(1), help(1), prs(1)

DIAGNOSTICS

Use **help**(1) for explanations.

BUGS

val can process up to 50 files on a single command line. Any number above 50 will produce a **core** dump.

NAME

vc – version control

SYNOPSIS

vc [**–a**] [**–t**] [**–c**char] [**–s**] [keyword=value ... keyword=value]

DESCRIPTION

The **vc** command copies lines from the standard input to the standard output under control of its arguments and control statements encountered in the standard input. In the process of performing the copy operation, user declared keywords may be replaced by their string value when they appear in plain text and/or control statements.

The copying of lines from the standard input to the standard output is conditional, based on tests (in control statements) of keyword values specified in control statements or as **vc** command arguments.

A control statement is a single line beginning with a control character, except as modified by the **–t** keyletter (see below). The default control character is colon (:), except as modified by the **–c** keyletter (see below). Input lines beginning with a backslash (\) followed by a control character are not control lines and are copied to the standard output with the backslash removed. Lines beginning with a backslash followed by a non-control character are copied in their entirety.

A keyword contains up to 9 alphanumerics; the first must be alphabetic. A value is any ASCII string that can be created with **ed**(1); a numeric value is an unsigned string of digits. Keyword values cannot contain blanks or tabs.

Replacement of keywords by values is done whenever a keyword surrounded by control characters is encountered on a version control statement. The **–a** keyletter (see below) forces replacement of keywords in all lines of text. An uninterpreted control character may be included in a value by preceding it with \. If a literal \ is desired, then it too must be preceded by \.

Keyletter Arguments

–a Forces replacement of keywords surrounded by control characters with their assigned value in *all* text lines and not just in **vc** statements.

–t All characters from the beginning of a line up to and including the first tab character are ignored for the purpose of detecting a control statement. If one is found, all characters up to and including the tab are discarded.

–cchar Specifies a control character to be used in place of the colon (:).

–s Silences warning messages (not error) that are normally printed on the diagnostic output.

Version Control Statements

:dcl keyword[, ..., keyword]

Used to declare keywords. All keywords must be declared.

:asg keyword=value

Used to assign values to keywords. An **asg** statement overrides the assignment for the corresponding keyword on the **vc** command line and all previous **asg**'s for that keyword. Keywords declared but not assigned

values have null values.

:if condition

.
.
.

:end

Used to skip lines of the standard input. If the condition is true, all lines between the *if* statement and the matching *end* statement are copied to the standard output. If the condition is false, all intervening lines are discarded, including control statements. Note that intervening *if* statements and matching *end* statements are recognized solely for the purpose of maintaining the proper *if-end* matching.

The syntax of a condition is as follows:

```
<cond>     ::= [ "not" ] <or>
<or>       ::= <and> | <and> "|" <or>
<and>      ::= <exp> | <exp> "&" <and>
<exp>      ::= "(" <or> ")" | <value> <op> <value>
<op>       ::= "=" | "!=" | "<" | ">"
<value>    ::= <arbitrary ASCII string> | <numeric string>
```

The available operators and their meanings are as follows:

=	Equal
!=	Not equal
&	And
|	Or
>	Greater than
<	Less than
( )	Used for logical groupings
not	May only occur immediately after the *if*, and when present, inverts the value of the entire condition

The > and < operate only on unsigned integer values (for example, : 012 > 12 is false). All other operators take strings as arguments (for example, : 012 != 12 is true). The precedence of the operators (from highest to lowest) is as follows:

= != > <	All of equal precedence
&	
|	

Parentheses may be used to alter the order of precedence.
Values must be separated from operators or parentheses by at least one blank or tab.

::text

Used for keyword replacement on lines that are copied to the standard output. The two leading control characters are removed, and keywords surrounded by control characters in text are replaced by their value before the line is copied to the output file. This action is independent of the **-a** keyletter.

:on

NAME

vc – version control

SYNOPSIS

vc [**–a**] [**–t**] [**–c**char] [**–s**] [keyword=value ... keyword=value]

DESCRIPTION

The **vc** command copies lines from the standard input to the standard output under control of its arguments and control statements encountered in the standard input. In the process of performing the copy operation, user declared keywords may be replaced by their string value when they appear in plain text and/or control statements.

The copying of lines from the standard input to the standard output is conditional, based on tests (in control statements) of keyword values specified in control statements or as **vc** command arguments.

A control statement is a single line beginning with a control character, except as modified by the **–t** keyletter (see below). The default control character is colon (:), except as modified by the **–c** keyletter (see below). Input lines beginning with a backslash (\) followed by a control character are not control lines and are copied to the standard output with the backslash removed. Lines beginning with a backslash followed by a non-control character are copied in their entirety.

A keyword contains up to 9 alphanumerics; the first must be alphabetic. A value is any ASCII string that can be created with **ed**(1); a numeric value is an unsigned string of digits. Keyword values cannot contain blanks or tabs.

Replacement of keywords by values is done whenever a keyword surrounded by control characters is encountered on a version control statement. The **–a** keyletter (see below) forces replacement of keywords in all lines of text. An uninterpreted control character may be included in a value by preceding it with \. If a literal \ is desired, then it too must be preceded by \.

Keyletter Arguments

–a Forces replacement of keywords surrounded by control characters with their assigned value in *all* text lines and not just in **vc** statements.

–t All characters from the beginning of a line up to and including the first tab character are ignored for the purpose of detecting a control statement. If one is found, all characters up to and including the tab are discarded.

–cchar Specifies a control character to be used in place of the colon (:).

–s Silences warning messages (not error) that are normally printed on the diagnostic output.

Version Control Statements

:dcl keyword[, ..., keyword]

Used to declare keywords. All keywords must be declared.

:asg keyword=value

Used to assign values to keywords. An **asg** statement overrides the assignment for the corresponding keyword on the **vc** command line and all previous **asg**'s for that keyword. Keywords declared but not assigned

values have null values.

:if condition

.
.
.

:end

Used to skip lines of the standard input. If the condition is true, all lines between the *if* statement and the matching *end* statement are copied to the standard output. If the condition is false, all intervening lines are discarded, including control statements. Note that intervening *if* statements and matching *end* statements are recognized solely for the purpose of maintaining the proper *if-end* matching.

The syntax of a condition is as follows:

```
<cond>     ::= [ "not" ] <or>
<or>       ::= <and> | <and> "|" <or>
<and>      ::= <exp> | <exp> "&" <and>
<exp>      ::= "(" <or> ")" | <value> <op> <value>
<op>       ::= "=" | "!=" | "<" | ">"
<value>    ::= <arbitrary ASCII string> | <numeric string>
```

The available operators and their meanings are as follows:

=	Equal
!=	Not equal
&	And
|	Or
>	Greater than
<	Less than
( )	Used for logical groupings
not	May only occur immediately after the *if*, and when present, inverts the value of the entire condition

The > and < operate only on unsigned integer values (for example, : 012 > 12 is false). All other operators take strings as arguments (for example, : 012 != 12 is true). The precedence of the operators (from highest to lowest) is as follows:

= != > <	All of equal precedence
&	
|	

Parentheses may be used to alter the order of precedence.
Values must be separated from operators or parentheses by at least one blank or tab.

::text

Used for keyword replacement on lines that are copied to the standard output. The two leading control characters are removed, and keywords surrounded by control characters in text are replaced by their value before the line is copied to the output file. This action is independent of the –a keyletter.

:on

:off Turn on or off keyword replacement on all lines.

:ctl char
Change the control character to char.

:msg message
Print the given message on the diagnostic output.

:err message
Print the given message followed by:

ERROR: err statement on line ... (915)

on the diagnostic output. **vc** halts execution and returns an exit code of 1.

SEE ALSO

ed(1), help(1)

DIAGNOSTICS

Use **help**(1) for explanations.

EXIT CODES:

0 – normal
1 – any error

NAME

vi – screen-oriented (visual) display editor based on ex

SYNOPSIS

vi [**–t** tag] [**–r** file] [**–L**] [**–w***n*] [**–R**] [**–x**] [**–C**] [**–c** command] file ...
view [**–t** tag] [**–r** file] [**–L**] [**–w***n*] [**–R**] [**–x**] [**–C**] [**–c** command] file ...
vedit [**–t** tag] [**–r** file] [**–L**] [**–w***n*] [**–R**] [**–x**] [**–C**] [**–c** command] file ...

DESCRIPTION

vi (visual) is a display-oriented text editor based on an underlying line editor **ex**(1). It is possible to use the command mode of **ex** from within **vi** and vice-versa. The visual commands are described on this manual page; how to set options (like automatically numbering lines and automatically starting a new output line when you type carriage return) and all **ex**(1) line editor commands are described on the **ex**(1) manual page.

When using **vi**, changes you make to the file are reflected in what you see on your terminal screen. The position of the cursor on the screen indicates the position within the file.

Invocation Options

The following invocation options are interpreted by **vi** (previously documented options are discussed in the **NOTES** section at the end of this manual page). The *file* argument indicates one or more files to be edited.

–t *tag* — Edit the file containing the *tag* and position the editor at its definition.

–r *file* — Edit *file* after an editor or system crash. (Recovers the version of *file* that was in the buffer when the crash occurred.)

–L — List all files saved as the result of an editor or system crash.

–w*n* — Set the default window size to *n*. This is useful when using the editor over a slow speed line.

–R — **Readonly** mode; the **readonly** flag is set, preventing accidental overwriting of the file.

–x — Encryption option; when used, **vi** simulates the **X** command of **ex**(1) and prompts the user for a key. This key is used to encrypt and decrypt text using the algorithm of **crypt**(1). The **X** command makes an educated guess to determine whether text read in is encrypted. The temporary buffer file is encrypted also, using a transformed version of the key typed in for the **–x** option. See the **WARNING** section and **crypt**(1).

–C — Encryption option; same as the **–x** option, except that **vi** simulates the **C** command of **ex**(1). The **C** command is like the **X** command of **ex**(1), except that all text read in is assumed to have been encrypted.

–c *command* — Begin editing by executing the specified editor *command* (usually a search or positioning command).

The **view** invocation is the same as **vi** except the **readonly** flag is set. The **vedit** invocation is intended for beginners. It is the same as **vi** except the **report** flag is set to 1, the **showmode** and **novice** flags are set, and **magic** is turned off, making **vi** easier to learn.

This version of **vi** supports multiple instances of a tag in a tags file. Multiple tags cause **vi** to enter a special mode and present a list of tags. The user can then scan through the list and select the desired tag using the folloing commands:

j move down in the list

k move up in the list

CR accept the displayed tag

n check next tag file

ESC abort the tags command

Other keystrokes will give an error indication.

vi Modes

Command Normal and initial mode. Other modes return to command mode upon completion. Use Esc (escape) to cancel a partial command.

Input Entered by setting any of these options: **a A i I o O c C s S R**. Arbitrary text may then be entered. Input mode is terminated normally with an Esc character; abnormally with an interrupt.

Last line Reading input for **:** / **?** or **!**; terminate by typing a carriage return; an interrupt cancels termination.

Command Summary

Return stands for carriage return and Esc stands for the escape key.

Sample Commands

← ↓ ↑ →	Arrow keys move the cursor
h j k l	Same as arrow keys
i*text* Esc	Insert *text*
cw*new* Esc	Change word to *new*
ea*s* Esc	Pluralize word (end of word; append **s**; escape from input state)
x	Delete a character
dw	Delete a word
dd	Delete a line
3dd	Delete 3 lines
u	Undo previous change
ZZ	Exit **vi**, saving changes
:q! Return	Quit, discarding changes
/*text* Return	Search for *text*
Ctrl-U, Ctrl-D	Scroll up or down
:*cmd* Return	Any **ex** or **ed** command

Counts Before vi Commands

Numbers may be typed as a prefix to some commands. They are interpreted in one of these ways:

Line/column number	z G |
Scroll amount	Ctrl-D Ctrl-U
Repeat effect	most of the rest

Interrupting, Canceling

Esc	End insert or incomplete cmd
Delete	(Delete or rubout) interrupts

File Manipulation

ZZ	If file modified, write and exit; else exit
:w Return	Write back changes
:w ! Return	Forced write, if permission originally not valid
:q Return	Quit
:q ! Return	Quit, discard changes
:e *name* Return	Edit file *name*
:e ! Return	Reedit, discard changes
:e + *name* Return	Edit, starting at end
:e +*n* Return	Edit starting at line *n*
:e # Return	Edit alternate file
:e ! # Return	Edit alternate file, discard changes
:w *name* Return	Write file *name*
:w ! *name* Return	Overwrite file *name*
:sh Return	Run shell, then return
: ! *cmd* Return	Run *cmd*, then return
:n Return	Edit next file in arglist
:n *args* Return	Specify new arglist
Ctrl-G	Show current file and line
:ta *tag* Return	Position cursor to *tag*

In general, any **ex** or **ed** command (such as **substitute** or **global**) may be typed, preceded by a colon and followed by a carriage return.

Positioning Within File

Ctrl-F	Forward screen
Ctrl-B	Backward screen
Ctrl-D	Scroll down half screen
Ctrl-U	Scroll up half screen
*n*G	Go to the beginning of the specified line (end default), where *n* is a line number
/*pat*	Next line matching *pat*
?*pat*	Previous line matching *pat*
n	Repeat last / or ? command
N	Reverse last / or ? command
/*pat*/+*n*	nth line after *pat*

?pat?–n	nth line before *pat*
]]	Next section/function
[[	Previous section/function
(	Beginning of sentence
)	End of sentence
{	Beginning of paragraph
}	End of paragraph
%	Find matching ( ) { or }

Adjusting the Screen

Ctrl-L	Clear and redraw window
Ctrl-R	Clear and redraw window if ^L is → key
z Return	Redraw screen with current line at window top
z– Return	Redraw screen with current line at window bottom
z. Return	Redraw screen with current line at window center
/pat/z– Return	Move *pat* line to bottom of window
z*n*. Return	Use *n*-line window
Ctrl-E	Scroll window down 1 line
Ctrl-Y	Scroll window up 1 line

Marking and Returning

``	Move cursor to previous context
''	Move cursor to first non-white space in line
m*x*	Mark current position with the ASCII lowercase letter *x*
`*x*	Move cursor to mark *x*
'*x*	Move cursor to first non-white space in line marked by *x*

Line Positioning

H	Top line on screen
L	Last line on screen
M	Middle line on screen
+	Next line, at first non-white
–	Previous line, at first non-white
Return	Return, same as +
↓ or j	Next line, same column
↑ or k	Previous line, same column

Character Positioning

^	First non white-space character
0	Beginning of line
$	End of line
h or →	Forward
l or ←	Backward
Ctrl-H	Same as ← (backspace)
space	Same as → (space bar)
f*x*	Find next *x*
F*x*	Find previous *x*

t*x*	Move to character prior to next *x*
T*x*	Move to character following previous *x*
;	Repeat last **f F t** or **T**
,	Repeat inverse of last **f F t** or **T**
*n***|**	Move to column *n*
%	Find matching ({) or }

Words, Sentences, Paragraphs

w	Forward a word
b	Back a word
e	End of word
)	To next sentence
}	To next paragraph
(	Back a sentence
{	Back a paragraph
W	Forward a blank-delimited word
B	Back a blank-delimited word
E	End of a blank-delimited word

Corrections During Insert

Ctrl-H	Erase last character (backspace)
Ctrl-W	Erase last word
Erase	Your erase character, same as Ctrl-H (backspace)
Kill	Your kill character, erase this line of input
\	Quotes your erase and kill characters
Esc	Ends insertion, back to command mode
Del	Interrupt, terminates insert mode
Ctrl-D	Backtab one character; reset left margin of *autoindent*
^Ctrl-D	Caret (^) followed by Ctrl-D; backtab to beginning of line; do not reset left margin of *autoindent*
0Ctrl-D	Backtab to beginning of line; reset left margin of *autoindent*
Ctrl-V	Quote non-printable character

Insert and Replace

a	Append after cursor
A	Append at end of line
i	Insert before cursor
I	Insert before first non-blank
o	Open line below
O	Open line above
r*x*	Replace single char with *x*
R*text* Esc	Replace characters

Operators

Operators are followed by a cursor motion and affect all text that would have been moved. For example, since **w** moves over a word, **dw** deletes the word that would be moved over. Double the operator (e.g., **dd**) to affect whole lines.

d	Delete
c	Change
y	Yank lines to buffer
<	Left shift
>	Right shift
!	Filter through command

Miscellaneous Operations

C	Change rest of line (**c$**)
D	Delete rest of line (**d$**)
s	Substitute characters (**cl**)
S	Substitute lines (**cc**)
J	Join lines
x	Delete characters (**dl**)
X	Delete characters before cursor (**dh**)
Y	Yank lines (**yy**)

Yank and Put

Put inserts the text most recently deleted or yanked; however, if a buffer is named (using the ASCII lowercase letters **a** - **z**), the text in that buffer is put instead.

3yy	Yank 3 lines
3yl	Yank 3 characters
p	Put back text after cursor
P	Put back text before cursor
"*x*p	Put from buffer *x*
"*x*y	Yank to buffer *x*
"*x*d	Delete into buffer *x*

Undo, Redo, Retrieve

u	Undo last change
U	Restore current line
.	Repeat last change
"*d*p	Retrieve *d*'th last delete

AUTHOR

vi and **ex** were developed by The University of California, Berkeley California, Computer Science Division, Department of Electrical Engineering and Computer Science.

FILES

/tmp	Default directory where temporary work files are placed; it can be changed using the **directory** option (see the **ex**(1) **set** command)
/usr/lib/terminfo/?/*	Compiled terminal description database

SEE ALSO

ed(1), edit(1), ex(1)

WARNINGS

The encryption options are available only in the United States.

Tampering with entries in */usr/lib/terminfo/?/** (for example, changing or removing an entry) can affect programs such as **vi**(1) that expect the entry to be present and correct. In particular, removing the "dumb" terminal may cause unexpected problems.

NOTES

Two options, although they continue to be supported, have been replaced in the documentation by options that follow the Command Syntax Standard (see **intro**(1)). A **–r** option that is not followed with an option-argument has been replaced by **–L** and *+command* has been replaced by **–c** *command*.

BUGS

Software tabs using Ctrl-T work only immediately after the *autoindent*.

Left and right shifts on intelligent terminals do not make use of insert and delete character operations in the terminal.

NAME

vmtune – tune system-wide virtual-memory parameters

SYNOPSIS

/etc/vmtune [**–f**] [*options*]

DESCRIPTION

vmtune provides a programmatic interface to the **vm_ctl**(2) system call, allowing display and tuning of system-wide virtual-memory parameters. With no arguments, **vmtune** prints the current contents of the **vmtune** structure in the kernel, which holds the tuning parameters. The decimal value of each field is printed next to the name of the field. All memory sizes are displayed and entered in Kbytes.

Options allow the modification of the parameters, either individually or as a group. Only the superuser may alter the setting of the parameters. Options come in both a short form (easy to type) and long form (easy to remember). Available options are as follows:

–r *minRS*
–minRS *minRS*
Minimum size of a process resident-set, in Kbytes. This value is silently enforced to be at least two, and limited by maxRS.

–R *maxRS*
–maxRS *maxRS*
Maximum size of a process resident-set, in Kbytes. This value is silently enforced to be at least two, and limited by a boot-time determined value that insures the resident-set can fit in physical memory.

–e *RSexecslop*
–RSexecslop *RSexecslop*
See *RSexecdiv* below.

–m *RSexecmult*
–RSexecmult *RSexecmult*
See *RSexecdiv* below.

–d *RSexecdiv*
–RSexecdiv *RSexecdiv*
The **exec**(2) system call determines the initial resident-set size of a process via:

((text + data + bss) * RSexecmult) / RSexecdiv + RSexecslop

where RSexecmult / RSexecdiv provides a rational fraction and all other values are in Kbytes.

–L *dirtylow*
–dirtylow *dirtylow*
Minimum size of the dirty-memory list, in Kbytes.

–H *dirtyhigh*
–dirtyhigh *dirtyhigh*
Maximum size of the dirty-memory list, in Kbytes. When the size of the dirty-list exceeds this, a system process begins writing pages to swap-space. Once this process starts, it won't stop until the size of the dirty-list is within the maximum pageout-size of *dirtylow*. The

maximum pageout size is currently 16 Kbytes.

-K *klout_look*
-klout_look *klout_look*
When the pageout process runs, it attempts to locate dirty pages that live in adjacent blocks on the swap device(s) to write with one operation (to minimize the number of I/O requests). *klout_look* specifies how deep into the dirty-list it will look (in kernel pages). A value of zero disables clustered pageouts.

-v *PFFvtime*
-PFFvtime *PFFvtime*
The number of clock ticks (10ms) of process virtual time between calls to the page-fault frequency algorithm. If zero, page-fault frequency adjustment of all processes is disabled. If non-zero, when the interval elapses the system normalizes the number of page-faults the process has taken to number of faults per second, and may adjust the resident-set size of the process up or down depending on the rate. This adjustment takes place on a per-process basis.

-D *PFFdecr*
-PFFdecr *PFFdecr*
The number of Kbytes to reduce the resident-set of the process if its page-fault rate is below *PFFlow*.

-l *PFFlow*
-PFFlow *PFFlow*
Page-fault frequency lower-limit (page-faults per second).

-I *PFFincr*
-PFFincr *PFFincr*
The number of Kbytes to increase the resident-set of the process if its page-fault rate is above *PFFhigh*.

-h *PFFhigh*
-PFFhigh *PFFhigh*
Page-fault frequency upper-limit (page-faults per second).

-s *minfree*
-minfree *minfree*
The lower-bound on average free memory (in Kbytes) before the swapper gets more aggressive about swapping out processes.

-S *desfree*
-desfree *desfree*
The swapper wants this much memory (in Kbytes) to be free on the average.

-M *maxdirty*
-maxdirty *maxdirty*
If the average size of the dirty-list (in Kbytes), the swapper becomes equally as aggressive as in the minfree case.

The operating system kernel does some sanity checking on the altered values, but improper or naive settings can adversely affect the performance and other behavior of the system. Because of this, **vmtune** normally dumps the values and asks for verification of the change. The **-f** flag forces the change with no

verification query.

SEE ALSO

getpagesize(2), vm_ctl(2)

BUGS

There should be guidelines on why, when, and how to alter these parameters. Naive setting of the parameters can adversely affect system performance.

NAME

volcopy – make literal copy of filesystem

SYNOPSIS

/etc/volcopy [options] **filesystem from_dev from_vol to_dev to_vol**

DESCRIPTION

volcopy makes a literal copy of the filesystem using a blocksize matched to the device. The following arguments are required:

filesystem **filesystem** is the mounted filesystem name (for example *root* or *u1*) of the filesystem being copied.

from_dev The **from_dev** is the physical disk section or tape (such as */dev/rdsk/zd0s2*, */dev/rmt/xt0*) which is being copied.

from_vol The **from_vol** is the physical volume name (such as **pk5**, **tp123**) from which the information is being read and should match the external label sticker. Label names are limited to six characters. **from_vol** may be – to use the existing volume name.

to_dev The **to_dev** is the physical disk section or tape (such as */dev/rdsk/zd0s2*, */dev/rmt/xt0*) which is being written on.

to_vol The **to_vol** is the physical volume name (such as **pk5**, or **tp123**) to which the information is being written and should match the external label sticker. Label names are limited to six characters. **to_vol** may be – to use the existing volume name.

The following flag arguments are optional:

–feet *tapelength*
Specifies the length of the tape in feet.

–bpi *density* Specifies the tape density in BPI. Only 800, 1600 and 6250 are allowed.

–reel *reel_number*
Specifies the reel number if not starting at the first reel.

–buf Specifies double buffering if the density is greater than or equal to 1600 bpi.

–a Invokes a verification sequence requiring a positive operator response instead of the standard 10 second delay before the copy is made

–s Invokes the `DEL if wrong` verification sequence. This is the default.

–y Assumes that the response to all questions is **yes**.

The program requests length and density information if it is not given on the command line or is not recorded on an input tape label. If the filesystem is too large to fit on one reel, **volcopy** prompts for additional reels. Labels of all reels are checked. Tapes may be mounted alternately on two or more drives. If **volcopy** is interrupted, it asks if the user wants to quit or wants a shell. In the latter case, the user can perform other operations (for example, **labelit**)

and return to **volcopy** by exiting the new shell.

If the file /etc/log/filesave.log exists, **volcopy** will write a dated entry to show the backup history of copied volumes. If the file does not exist, a warning is generated.

In summary, **from_dev** and **from_vol** are the device and volume from which the copy of the filesystem is being extracted. **to_dev** and **to_vol** are the target device and volume.

EXAMPLES

In the following example, the *root* filesystem is being copied from one disk to another:

```
volcopy root /dev/rdsk/zd0s0 pk5 /dev/rdsk/zd1s2 pk12
```

In this example, the *u3* filesystem is being copied from a disk to a 2400 foot magnetic tape at 3250 bpi:

```
volcopy -feet 2400 -bpi 6250 u3 /dev/rdsk/sd1s3 pk1 \
    /dev/rmt/xt0 tp123
```

In the following example, a disk is being restored from a magnetic tape. Using a hyphen for the to and from volume names tells **volcopy** to use the label that is already on the tape.

```
volcopy u5 /dev/rmt/xt0 - /dev/rdsk/zd1s3 -
```

FILES

/etc/log/filesave.log A record of filesystems/volumes copied

SEE ALSO

sh(1), labelit(1M)

WARNINGS

volcopy does not support tape-to-tape copying. Use **dd**(1) for tape-to-tape copying.

NAME

vsig – synchronize a co-process with the controlling FMLI object

SYNOPSIS

vsig

DESCRIPTION

The **vsig** executable sends a SIGUSR2 signal to the controlling FMLI process. This signal causes FMLI to execute the FMLI command **checkworld**, causing all displayed objects with a `reread` descriptor evaluating to TRUE to be reread.

EXAMPLES

The following is a segment of a shell program:

```
echo "Sending this string to an FMLI process"
vsig
```

The **vsig** command flushes the output buffer before it sends the SIGUSR2 signal to make sure the string is actually in the pipe created by the **cocreate** built-in.

SEE ALSO

coproc(1F), signal(2)

NAME

wait – await completion of process

SYNOPSIS

wait [*n*]

DESCRIPTION

Wait for your background process having process ID *n* and report its termination status. If *n* is omitted, all your shell's currently active background processes are waited for and the return code will be zero.

The shell itself executes **wait** without creating a new process.

SEE ALSO

sh(1), ksh(1), csh(1)

DIAGNOSTICS

If you receive the error message *cannot fork, too many processes*, try using the **wait**(1) command to clean up your background processes. If this doesn't help, the system process table is probably full or you have too many active foreground processes. (There is a limit to the number of process IDs associated with your login and to the number the system can track.)

BUGS

Not all processes of a 3- or more-stage pipeline are children of the shell, and thus cannot be waited for.

If *n* is not an active process ID, all your shell's currently active background processes are waited for and the return code will be zero.

NAME

wall – write to all users

SYNOPSIS

/etc/wall

DESCRIPTION

wall reads its standard input until an end-of-file. It then sends the message to all currently logged-in users preceded by the following line:

```
Broadcast Message from...
```

It is used to warn all users, typically prior to shutting down the system.

The sender must be superuser to override any protections the users may have invoked (see **mesg**(1)).

FILES

/dev/tty*

SEE ALSO

mesg(1), write(1)

DIAGNOSTICS

Cannot send to ... The open on a user's tty file failed.

NAME

wc – word count

SYNOPSIS

wc [**–lwc**] [names]

DESCRIPTION

wc counts lines, words, and characters in the named files, or in the standard input if no *names* appear. It also keeps a total count for all named files. A word is a maximal string of characters delimited by spaces, tabs, or newlines.

The options **l**, **w**, and **c** may be used in any combination to specify that a subset of lines, words, and characters are to be reported. The default is **–lwc**.

When *names* are specified on the command line, they will be printed along with the counts.

NAME

what – identify SCCS files

SYNOPSIS

what [**–s**] files

DESCRIPTION

what searches the given files for all occurrences of the pattern that **get**(1) substitutes for %Z% (this is **@(#)** at this printing) and prints out what follows until the first ", >, newline, \, or null character. For example, if the C program in file *f.c* contains this line:

```
#ident  "@(#)identification information"
```

and *f.c* is compiled to yield *f.o* and *a.out*, then the command

what f.c f.o a.out

will print the following:

f.c:
    identification information

f.o:
    identification information

a.out:
    identification information

what is intended to be used in conjunction with the command **get**(1), which automatically inserts identifying information, but it can also be used where the information is inserted manually. Only one option exists:

–s Quit after finding the first occurrence of pattern in each file.

SEE ALSO

get(1), help(1)

DIAGNOSTICS

Exit status is 0 if any matches are found, otherwise 1. Use **help**(1) for explanations.

BUGS

It is possible that an unintended occurrence of the pattern **@(#)** could be found just by chance, but this causes no harm in nearly all cases.

NAME

whatis – describe what a command is

SYNOPSIS

whatis command ...

DESCRIPTION

whatis looks up a command and displays the header line from the manual section. You can then run the **man**(1) command to get more information. If the line starts `name(section) ...`, enter **man section name** to get the documentation for it. For example, you can enter **whatis ed** and then **man 1 ed** to get the manual page.

whatis is equivalent to the **man –f** command.

FILES

/usr/catman/whatis Data base

SEE ALSO

man(1), apropos(1), makewhatis(1M)

NAME

who – who is on the system

SYNOPSIS

who [**–uTlHqpdbrtas**] [**-n** *x*] [file]

who am i

who am I

DESCRIPTION

who can list the user's name, terminal line, login time, elapsed time since activity occurred on the line, and the process-ID of the command interpreter (shell) for each current user. It examines the */etc/utmp* file at login time to obtain its information. If *file* is given, that file (which must be in **utmp**(4) format) is examined. Usually, *file* will be */etc/wtmp*, which contains a history of all the logins since the file was last created.

who with the **am i** or **am I** option identifies the invoking user.

The general format for output is as follows:

name [state] line time [idle] [pid] [comment] [exit]

The *name*, *line*, and *time* information is produced by all options except **–q**; the *state* information is produced only by **–T**; the *idle* and *pid* information is produced only by **–u** and **–l**; and the *comment* and *exit* information is produced only by **–a**. The information produced for **–p**, **–d**, and **–r** is explained during the discussion of each option, below.

With options, **who** can list logins, logoffs, reboots, and changes to the system clock, as well as other processes spawned by the **init** process. These options are as follows:

–u This option lists only those users who are currently logged in. The *name* is the user's login name. The *line* is the name of the line as found in the directory */dev*. The *time* is the time that the user logged in. The *idle* column contains the number of hours and minutes since activity last occurred on that particular line. A dot (.) indicates that the terminal has seen activity in the last minute and is therefore "current". If more than 24 hours have elapsed or the line has not been used since boot time, the entry is marked **old**. This field is useful when trying to determine whether a person is working at the terminal. The *pid* is the process-ID of the user's shell. The *comment* is the comment field associated with this line as found in */etc/inittab* (see **inittab**(4)). This can contain information about where the terminal is located, the telephone number of the dataset, type of terminal if hard-wired, etc.

–T This option is the same as the **–s** option, except that the *state* of the terminal line is printed. The *state* describes whether someone else can write to that terminal. A **+** appears if the terminal is writable by anyone; a **–** appears if it is not. **root** can write to all lines having a **+** or a **–** in the *state* field. If a bad line is encountered, a **?** is printed.

–l This option lists only those lines on which the system is waiting for someone to login. The *name* field is **LOGIN** in such cases. Other fields are the same as for user entries except that the *state* field does not exist.

-H This option will print column headings above the regular output.

-q This is a quick **who**, displaying only the names and the number of users currently logged on. When this option is used, all other options are ignored.

-p This option lists any other process which is currently active and has been previously spawned by **init**. The *name* field is the name of the program executed by **init** as found in */etc/inittab*. The *state*, *line*, and *idle* fields have no meaning. The *comment* field shows the *id* field of the line from */etc/inittab* that spawned this process. See **inittab**(4).

-d This option displays all processes that have expired and not been respawned by **init**. The *exit* field appears for dead processes and contains the termination and exit values (as returned by **wait**(2)), of the dead process. This can be useful in determining why a process terminated.

-b This option indicates the time and date of the last reboot.

-r This option indicates the current *run-level* of the **init** process. In addition, it produces the process termination status, process id, and process exit status (see **utmp**(4)) under the *idle*, *pid*, and *comment* headings, respectively.

-t This option indicates the last change to the system clock (via the **date**(1) command) by **root**. See **su**(1).

-a This option processes */etc/utmp* or the named *file* with all options turned on.

-s This option is the default and lists only the *name*, *line*, and *time* fields.

-n *x* This option takes a numeric argument, *x*, which specifies the number of users to display per line. *x* must be at least **1**. The **-n** option must be used with **-q**.

Note to the superuser: after a shutdown to the single-user state, **who** returns a prompt; the reason is that since */etc/utmp* is updated at login time and there is no login in single-user state, **who** cannot report accurately on this state. **who am i**, however, returns the correct information.

FILES

/etc/utmp
/etc/wtmp
/etc/inittab

SEE ALSO

date(1), login(1), mesg(1), init(1M), su(1M), wait(2), inittab(4), utmp(4)

NAME

whodo – who is doing what

SYNOPSIS

/etc/whodo

DESCRIPTION

whodo produces formatted and dated output from information in the */etc/utmp* and */etc/ps_data* files.

The display is headed by the date, time and machine name. For each user logged in, device name, user-id and login time is shown, followed by a list of active processes associated with the user-id. The list includes the device name, process-id, cpu minutes and seconds used, and process name.

EXAMPLES

The command:

whodo

produces a display like this:

```
Fri Jun  9 17:52:14 1989
elm229

syscon       root     17:50
    syscon       13303    0:00 sh

ttyGA/GAAB  dale      8:27
    ttyGA/GAAB  29083    0:05 ksh
    ttyGA/GAAB  15194    0:00 vi
    ttyGA/GAAB  15196    0:00 whodo

ttyGA/GAAF  joe      14:59
    ttyGA/GAAF  12669    0:00 ksh
    ttyGA/GAAF  12797    0:12 ksh
    ttyGA/GAAF  15169    0:01 menu
```

FILES

/etc/passwd
/etc/ps_data
/etc/utmp

SEE ALSO

ps(1), who(1)

NAME

write – write to another user

SYNOPSIS

write user [line]

DESCRIPTION

write copies lines from your terminal to that of another user. When first called, it sends the following message to the person you want to talk to:

```
Message from yourname (tty??/????) [date] ...
```

When it has successfully completed the connection, it also sends two bells to your own terminal to indicate that what you are typing is being sent.

The recipient of the message should write back at this point. Communication continues until an end of file is read from the terminal, an interrupt is sent, or the recipient has executed **mesg n**. At that point **write** writes **EOT** on the other terminal and exits.

If you want to write to a user who is logged in more than once, the *line* argument may be used to indicate which line or terminal to send to (such as *ttyGA/AAAA*). Otherwise, the first writable instance of the user found in */etc/utmp* is assumed and the following message is posted:

```
user is logged on more than one place.
You are connected to "terminal".
Other locations are:
terminal
```

Permission to write may be denied or granted by use of the **mesg**(1) command. Writing to others is normally allowed by default. Certain commands such as **pr**(1) disallow messages in order to prevent interference with their output. However, if the user has superuser permissions, messages can be forced onto a write-inhibited terminal.

If the character **!** is found at the beginning of a line, **write** calls the shell to execute the rest of the line as a command.

The following protocol is suggested for using **write**: when you first **write** to another user, wait for them to **write** back before starting to send. Each person should end a message with a distinctive signal (such as **(o)** for "over") so that the other person knows when to reply. The signal **(oo)** (for "over and out") is suggested when conversation is to be terminated.

FILES

/etc/utmp	To find user
/bin/sh	To execute !

SEE ALSO

mail(1), mesg(1), pr(1), sh(1), who(1)

DIAGNOSTICS

user is not logged on
: The person you are trying to **write** to is not logged on.

Permission denied
: The person you are trying to **write** to denies that permission (with **mesg**).

Warning: cannot respond, set mesg –y
Your terminal is set to **mesg n** and the recipient cannot respond to you.

Can no longer write to user
The recipient denied permission (*mesg n*) after you started writing.

NAME

xargs – construct argument list(s) and execute command

SYNOPSIS

xargs [flags] [command [initial-arguments]]

DESCRIPTION

xargs combines the fixed *initial-arguments* with arguments read from standard input to execute the specified *command* one or more times. The number of arguments read for each *command* invocation and the manner in which they are combined are determined by the flags specified.

command, which may be a shell file, is searched for by using one's **$PATH**. If *command* is omitted, **/bin/echo** is used.

Arguments read in from standard input are defined to be contiguous strings of characters delimited by one or more blanks, tabs, or newlines; empty lines are always discarded. Blanks and tabs may be embedded as part of an argument if escaped or quoted. Characters enclosed in quotes (single or double) are taken literally, and the delimiting quotes are removed. Outside of quoted strings, a backslash (\) will escape the next character.

Each argument list is constructed starting with the *initial-arguments* followed by some number of arguments read from standard input (Exception: see **–i** flag). Flags **–i**, **–l**, and **–n** determine how arguments are selected for each command invocation. When none of these flags are coded, the *initial-arguments* are followed by arguments read continuously from standard input until an internal buffer is full. Then *command* is executed with the accumulated args. This process is repeated until there are no more args. When there are flag conflicts (for example, **–l** vs. **–n**), the last flag has precedence.

Flag values are as follows:

–l*number* *command* is executed for each non-empty *number* lines of arguments from standard input. The last invocation of *command* will be with fewer lines of arguments if fewer than *number* remain. A line is considered to end with the first newline unless the last character of the line is a blank or a tab. A trailing blank/tab signals continuation through the next non-empty line. If *number* is omitted, 1 is assumed. Option **–x** is forced.

–i*replstr* Insert mode: *command* is executed for each line from standard input, taking the entire line as a single arg, inserting it in *initial-arguments* for each occurrence of *replstr*. A maximum of 5 arguments in *initial-arguments* may each contain one or more instances of *replstr*. Blanks and tabs at the beginning of each line are thrown away. Constructed arguments may not grow larger than 255 characters, and option **–x** is also forced. {} is assumed for *replstr* if not specified.

–n*number* Execute *command* using as many standard input arguments as possible, up to *number* arguments maximum. Fewer arguments will be used if their total size is greater than *size* characters, and for the last invocation if there are fewer than *number* arguments remaining. If option **–x** is also coded, each *number* arguments must fit in the *size* limitation, else **xargs** terminates execution.

–t Trace mode: The *command* and each constructed argument list are echoed to file descriptor 2 just prior to their execution.

–p Prompt mode: The user is asked whether to execute *command* each invocation. Trace mode (**–t**) is turned on to print the command instance to be executed, followed by a **?...** prompt. A reply of **y** (optionally followed by anything) will execute the command; anything else, including just a carriage return, skips that particular invocation of *command*.

–x Causes **xargs** to terminate if any argument list would be greater than *size* characters; **–x** is forced by the options **–i** and **–l**. When none of the options **–i**, **–l**, or **–n** are coded, the total length of all arguments must be within the *size* limit.

–s*size* The maximum total size of each argument list is set to *size* characters; *size* must be a positive integer less than or equal to 470. If **–s** is not coded, 470 is taken as the default. Note that the character count for *size* includes one extra character for each argument and the count of characters in the command name.

–e*eofstr* *eofstr* is taken as the logical end-of-file string. Underbar (_) is assumed for the logical **EOF** string if **–e** is not coded. The value **–e** with no *eofstr* coded turns off the logical EOF string capability (underbar is taken literally). **xargs** reads standard input until either end-of-file or the logical EOF string is encountered.

xargs will terminate if it receives a return code of **–1** from, or if it cannot execute, *command*. When *command* is a shell program, it should explicitly **exit** (see **sh**(1)) with an appropriate value to avoid accidentally returning with **–1**.

EXAMPLES

The following example moves all files from directory $1 to directory $2 and echos each move command just before doing it:

```
ls $1 | xargs -i -t mv $1/{} $2/{}
```

The following example combines the output of the parenthesized commands onto one line, which is then echoed to the end of file *log*:

```
(logname; date; echo $0 $*) | xargs >>log
```

The user is asked which files in the current directory are to be archived and archives them into *arch* one at a time (example 1), or many at a time (example 2).

```
Example 1) ls | xargs -p -1 ar r arch
Example 2) ls | xargs -p -l | xargs ar r arch
```

The following example executes **diff**(1) with successive pairs of arguments originally typed as shell arguments:

```
echo $* | xargs -n2 diff
```

SEE ALSO

sh(1)

NAME

yacc – yet another compiler-compiler

SYNOPSIS

yacc [**–vdlt**] grammar

DESCRIPTION

The **yacc** command converts a context-free grammar into a set of tables for a simple automaton which executes an LR(1) parsing algorithm. The grammar may be ambiguous; specified precedence rules are used to break ambiguities.

The output file, *y.tab.c*, must be compiled by the C compiler to produce a program **yyparse**. This program must be loaded with the lexical analyzer program, **yylex**, as well as **main** and **yyerror**, an error handling routine. These routines must be supplied by the user; **lex**(1) is useful for creating lexical analyzers usable by **yacc**.

If the **–v** flag is given, the file *y.output* is created containing a description of the parsing tables and a report on conflicts generated by grammar ambiguities.

If the **–d** flag is used, the file *y.tab.h* is generated with the *#define* statements that associate the **yacc**-assigned "token codes" with the user-declared "token names". This allows source files other than *y.tab.c* to access the token codes.

If the **–l** flag is given, the code produced in *y.tab.c* does not contain any *#line* constructs. This should only be used after the grammar and the associated actions are fully debugged.

Runtime debugging code is always generated in *y.tab.c* under conditional compilation control. By default, this code is not included when *y.tab.c* is compiled. However, when **yacc**'s **–t** option is used, this debugging code is compiled by default. Independent of whether the **–t** option was used, the runtime debugging code is under the control of YYDEBUG, a preprocessor symbol. If YYDEBUG has a non-zero value, debugging code is included. If its value is zero, the code is not included. The size and execution time of a program produced without the runtime debugging code is smaller and slightly faster.

FILES

y.output	
y.tab.c	
y.tab.h	Defines for token names
yacc.tmp, yacc.debug, yacc.acts	Temporary files
/usr/lib/yaccpar	Parser prototype for C programs

SEE ALSO

lex(1)

DIAGNOSTICS

The number of reduce-reduce and shift-reduce conflicts is reported on standard error; a more detailed report is found in the *y.output* file. Similarly, if some rules are not reachable from the start symbol, this is also reported.

WARNINGS

Because filenames are fixed, at most one **yacc** process can be active in a given directory at a given time.